Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

IE – 0502 Proyecto Eléctrico

Tutor de control interactivo para Internet desarrollado en Ch

Por:

Alejandro Mora Sojo

Ciudad Universitaria Rodrigo Facio
Diciembre del 2006

Tutor de control interactivo para Internet desarrollado en Ch

Por:

Alejandro Mora

Sometido a la Escuela de Ingeniería Eléctrica
de la Facultad de Ingeniería
de la Universidad de Costa Rica
como requisito parcial para optar por el grado de:

BACHILLER EN INGENIERÍA ELÉCTRICA

Aprobado por el Tribunal:

_____
Ing. Víctor M. Alfaro Ruíz, M.Sc.
Profesor Guía

_____
Ing. Federico Ruiz Ugalde
Profesor lector

_____
Ing. Andrés Díaz Soto
Profesor lector

DEDICATORIA

A mis padres quienes me han impulsado siempre a alcanzar mis metas.

A mis hermanos por inspirarme a dar siempre lo mejor de mí.

RECONOCIMIENTOS

A mis profesores quienes me han enseñado el camino al conocimiento y a mis amigos, compañeros en este recorrido.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS vii

NOMENCLATURA viii

RESUMEN ix

CAPÍTULO 1: Introducción 1

1.1 Objetivos 1
1.1.1 Objetivo general 1
1.1.2 Objetivos específicos 1
1.2 Justificación del tema 2
1.3 Metodología 3

CAPITULO 2: Tutores de control disponibles en Internet 5

2.1 Sitio de John Hopkins University (E.U.A.) 5
2.1.1 LTI arcade [4] 5
2.1.2 Exploring the s-plane [5] 6
2.1.3 Bode servo analysis [6] 7
2.1.4 Sense and sensitivities [7] 8
2.2 Sitio de Newcastle University (Australia) [8] 9
2.3 Sitio de la Universidad de Exeter (Reino Unido) [9] 12
2.4 Sitio de la Universidad de Tennesse en Chatannogga (E.U.A.) [10] 13
2.5 Sitio de la empresa TechTeach [11] 14
2.6 Página personal de John Shaw [12] 15
2.7 Sitio de la Universidad de Bochum en Ruhr (Alemania) [13] 16
2.8 Sitio de la Escuela Superior de Ingenieros de Bilbao (España) [14] 17
2.9 Softintegration [15] 18

CAPÍTULO 3: Lenguaje de programación Ch 20

3.1 Características generales 20
3.2 Capacidad para resolver problemas de control automático [3] 22
3.3 Capacidad de desarrollo de sitios Web interactivos [2] 23

CAPÍTULO 4. Tutor interactivo para control automático 25

4.1 Respuesta de la planta a una entrada escalón 26
4.2 Identificación de modelos 27

4.3 Herramientas clásicas para el análisis de sistemas de control 29
4.4 Controladores P, PI y PID.....30
4.5 Sintonización de controladores.....31
CAPÍTULO 5: Conclusiones y recomendaciones 34
BIBLIOGRAFÍA.....36
APÉNDICES 38
Guía del Programador.....38
Código Fuente.....56

ÍNDICE DE FIGURAS

Figura 2.1. Vista en pantalla de Signal Arcade....6
Figura 2.2 Vista en pantalla de Exploring the s-plane....7
Figura 2.3 Sistema analizado en Bode Servo analysis [6]....8
Figura 2.4 Vista en pantalla de Bode Servo Analysis....8
Figura 2.5 Modelo del sistema analizado en Sense and Sensitivities [7]....9
Figura 2.6 Vista en pantalla de Sense and Sensitivities....9
Figura 2.7 Vista en pantalla del sitio de la Universidad de Newcastle....10
Figura 2.8 Vista en pantalla del sitio de la Universidad de Exeter....13
Figura 2.9 Sitio de la Universidad de Tennesse en Chatannogga....14
Figura 2.10 Vista en pantalla del sitio de TechTeach....15
Figura 2.11 Vista en pantalla del sitio de la Universidad de Bochum....17
Figura 3.1 Relación de Ch con otros lenguajes de programación [1]....21
Figura 4.1 Pantalla de entrada de datos del módulo para respuesta de la planta....26
Figura 4.2 Curvas generadas por el módulo de respuesta de la planta....27
Figura 4.3 Pantalla de entrada del módulo para identificación de modelos....28
Figura 4.4 Respuesta del módulo de identificación de modelos....28
Figura 4.5 Pantalla de entrada del módulo para análisis de estabilidad....29
Figura 4.6 Respuesta del módulo para análisis de estabilidad....30
Figura 4.7 Pantalla de entrada del módulo de controladores....31
Figura 4.8 Respuesta del módulo para modos de control....31
Figura 4.9 Pantalla de entrada para el módulo de sintonización de controladores....33
Figura 4.10 Respuesta del módulo de sintonización....33

NOMENCLATURA

Java	Lenguaje de programación orientado a objetos desarrollado de Sun Microsystems
Controlador P	Controlador Proporcional
Controlador PI	Controlador Proporcional Integral
Controlador PID	Controlador Proporcional Integral Derivativo
SISO	Sistemas de una entrada y una salida
C Shell	Procesador de commandos de línea desarrollado para Unix
Html	Lenguaje usado para crear las páginas Web

RESUMEN

Considerando los requerimientos de software de los estudiantes del curso de sistemas de control automático y en vista de los obstáculos que estos encuentran para tener acceso a estas herramientas debido a los costos de las licencias y requerimientos de instalación y de hardware, se emprendió el proyecto de desarrollar una herramienta de análisis de sistemas de control que estuviera disponible en Internet de manera gratuita y que no tuviera muchos requerimientos técnicos.

Previo al desarrollo de la herramienta que aquí se presenta se realizó una búsqueda en Internet para evaluar las herramientas existentes y determinar si estas se adecuan satisfactoriamente al contenido del curso de Sistemas de Control IE-431.

Seguidamente y por medio de un estudio de sus características, se determinó que el lenguaje Ch contaba con las herramientas adecuadas para desarrollar un tutor interactivo para el curso, que permitiera a los estudiantes realizar las labores propias de este de una manera rápida y ágil vía Internet y sin la necesidad de instalar programas.

El fondo del proyecto consistió en el propio desarrollo de la herramienta, donde se detectaron algunas limitaciones del lenguaje Ch para el análisis de Sistemas de Control y finalmente se procedió elaborar el manual de usuario y la guía del programador, así como su instalación en uno de los servidores de la Escuela de Ingeniería Eléctrica.

CAPÍTULO 1: Introducción

1.1 Objetivos

1.1.1 Objetivo general

- Desarrollar un sistema tutor de control automático interactivo, con los temas del curso Sistemas de Control, para ser utilizado por Internet, programado en el lenguaje Ch.

1.1.2 Objetivos específicos

- Analizar el funcionamiento y capacidad de los tutores de control disponibles en Internet
- Estudiar las características generales del lenguaje de programación Ch, su capacidad para resolver problemas de control automático y desarrollar sitios Web interactivos
- Seleccionar los temas de control a ser tratados en el tutor y la organización del mismo, de conformidad con el temario del curso IE-431 Sistemas de control
- Desarrollar el tutor en el lenguaje Ch, instalarlo en el servidor Web de la Escuela y probar su funcionamiento
- Elaborar un manual de usuario para el mismo

1.2 Justificación del tema

El concepto del proyecto se concibió para facilitar a los estudiantes del curso de control automático, las labores propias de este tales como la realización de tareas y el estudio individual. Actualmente las herramientas de software con las que cuentan los estudiantes presentan una serie de desventajas, algunas de estas son:

- Alto costo de licencias
- Grandes requerimientos de espacio en disco y memoria
- Necesidad de instalación
- Poca documentación en español

La aplicación desarrollada busca solucionar esas desventajas incorporando las siguientes características:

- Gratuito
- Disponibilidad desde cualquier punto con conexión a Internet
- Adecuado al contenido del curso
- No requiere espacio en disco duro

Asimismo al ser una herramienta disponible vía Internet a estudiantes y profesores en todo el mundo, refleja el interés institucional de vincularse y realizar aportes a la comunidad científica. De igual manera el proyecto busca disminuir la piratería de programas informáticos.

1.3 Metodología

Para el desarrollo del proyecto se dividió el trabajo en cinco etapas correspondientes a los cinco objetivos específicos definidos previamente.

En la primera etapa, se procedió a buscar en Internet sitios relacionados a la teoría de control, tanto moderna como clásica, que incluyeran algún tipo de tutor o la posibilidad de realizar experimentos, o ejecutar simulaciones o pruebas de manera interactiva. La búsqueda se realizó utilizando los portales de búsqueda Yahoo! (www.yahoo.com) y Google (www.google.com) utilizando las palabras claves "*Control system tutorial*", "*Control system interactive*", "*Control system*", "sistemas de control interactivo", "tutorial sistemas de control".

La segunda etapa del proyecto consistió en el estudio del lenguaje Ch y su capacidad para resolver problemas de control automático y desarrollar sitios Web interactivos. Durante esta etapa se estudió principalmente la documentación de Ch, que se encuentra en el sitio www.softintegration.com [1], donde se distribuye de manera gratuita el lenguaje Ch. Esta documentación incluye la guía de instalación, la guía de referencia y la guía para el usuario las cuales se encuentran en el paquete de instalación del lenguaje. Adicionalmente se estudió la guía del conjunto de herramientas [2] de CGI ("*Common gateway interface*") de Ch y la guía del conjunto de herramientas [3] de sistemas de control, donde se encontró, respectivamente, la información necesaria para el desarrollo de sitios Web interactivos y la documentación correspondiente a las herramientas para resolver problemas de control automático.

La etapa siguiente consistió en la selección de los temas a incluir entre los módulos interactivos. Para esto se comparó el temario del curso de Sistemas de Control IE-0431 del segundo semestre del 2006 con las herramientas disponibles en el conjunto de herramientas para sistemas de control del lenguaje Ch. Luego de este análisis y considerando el peso de los temas con base en preguntas de exámenes de periodos anteriores, se estableció una lista de módulos a ser incluidos.

La siguiente fase del proyecto consistió en la creación de los módulos interactivos. Previo a esta etapa se probó la funcionalidad de Ch en un servidor Web de la Escuela. Para la creación de los módulos se utilizó la versión profesional de Ch, disponible de manera gratuita en www.softintegration.com y se usó el ambiente de desarrollo integrado ChSciTE 1.5 que permite editar los archivos y ejecutarlos de una manera simple para el programador, ya que permite observar donde se encuentran los errores. Luego de la creación de los módulos se procedió a la instalación de los mismos en los servidores de la escuela.

La fase final del proyecto consistió en realizar la documentación del proyecto. Se redactó una guía para el programador, para facilitar futuras extensiones y el mantenimiento de la plataforma. Asimismo, se elaboró una guía para el usuario que estará disponible en el sitio y que servirá de referencia al usuario mientras visita el sitio.

CAPITULO 2: Tutores de control disponibles en Internet

El presente capítulo muestra los resultados obtenidos al realizar una búsqueda en Internet de herramientas para el análisis de sistemas de control automático. Se analizan las principales características de cada uno de los sitios y los hallazgos obtenidos se utilizaron como referencia, para seleccionar los contenidos de los módulos desarrollados como parte del tutor.

2.1 Sitio de John Hopkins University (E.U.A.)

2.1.1 LTI arcade [4]

Permite al usuario observar la salida del sistema ante una entrada que el mismo dibuje con el apuntador ("*mouse*"). Los sistemas estudiados se limitan a sistemas de segundo y primer orden predefinidos en la aplicación. Aparte de la salida del servomecanismo con este tipo de función de transferencia se muestran ventanas que muestran la respuesta al impulso, al escalón y el diagrama de Bode para dichos sistemas. También muestra la relación entrada salida expresada como convolución, ecuación diferencial y función de transferencia (transformada de Laplace).

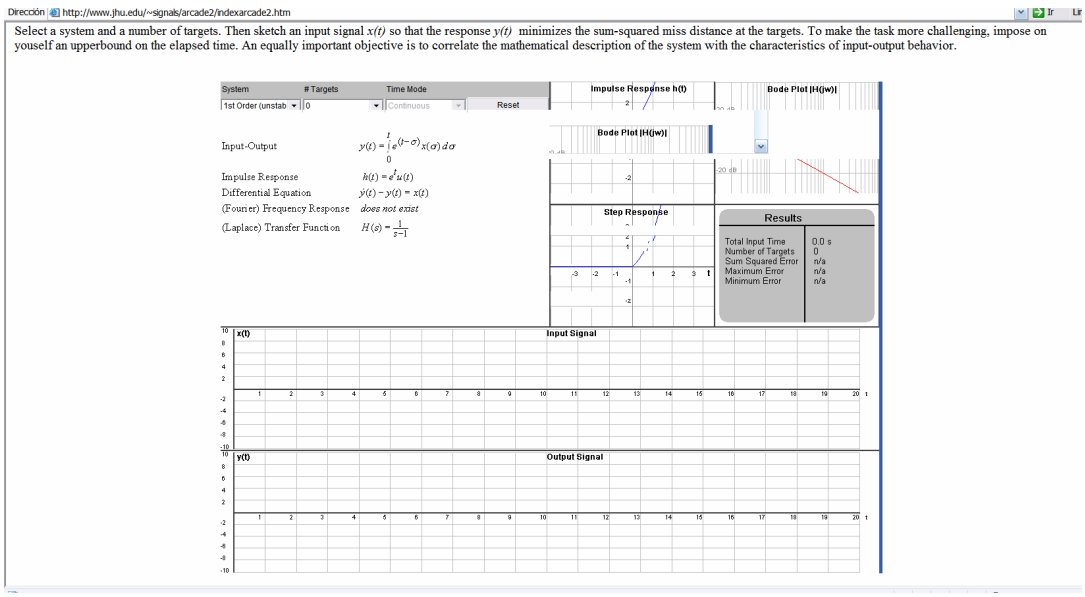

Figura 2.1. Vista en pantalla de Signal Arcade

2.1.2 Exploring the s-plane [5]

Esta aplicación desarrollada en Java permite a quienes lo naveguen, observar la respuesta al escalón de un sistema lineal invariante en el tiempo. Los polos y ceros de la función de transferencia del sistema, son gráficamente definidos por el usuario al marcarlos con el apuntador sobre el plano complejo. Permite variar la ganancia del sistema. Además determina si el sistema es inestable.

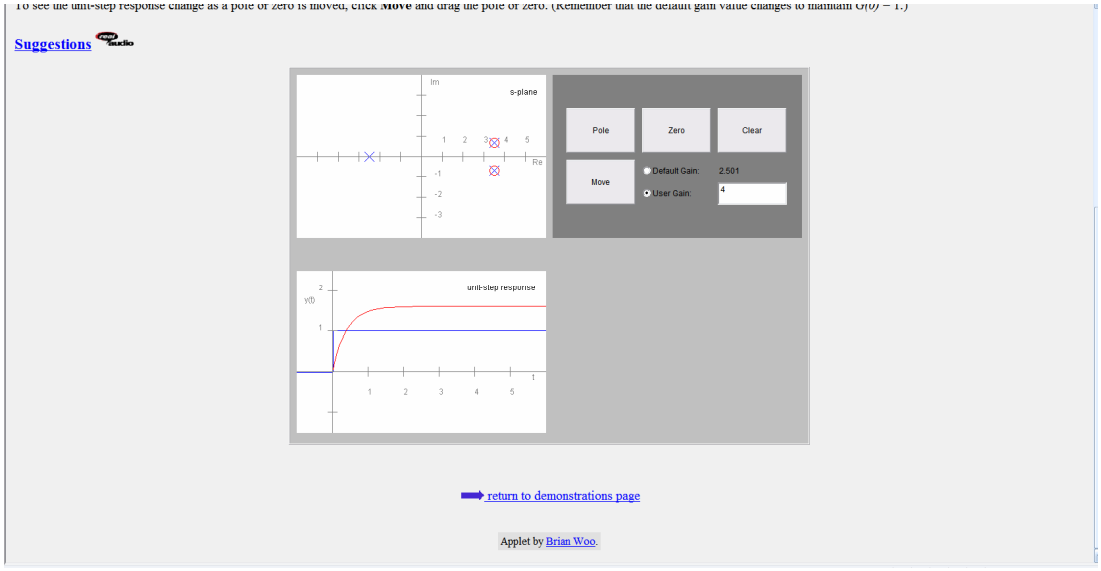

Figura 2.2 Vista en pantalla de Exploring the s-plane

2.1.3 Bode servo analysis [6]

Esta aplicación permite observar la respuesta al escalón de un sistema de tercer orden realimentado, a cuyo lazo de realimentación se le agrega el ruido generado por un sensor de medición. Se le permite al usuario decidir si la ganancia al ruido es nula, alta o baja y modificar el gráfico de Bode de ganancia de lazo abierto variando la frecuencia de esquina. Utilizando esta información, la aplicación genera la forma de la respuesta de lazo cerrado a una entrada escalón unitario.

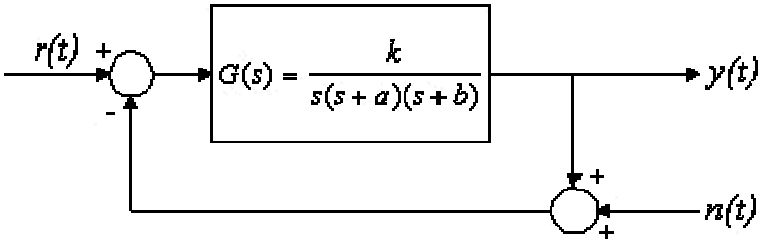

Figura 2.3 Sistema analizado en Bode Servo analysis [6]

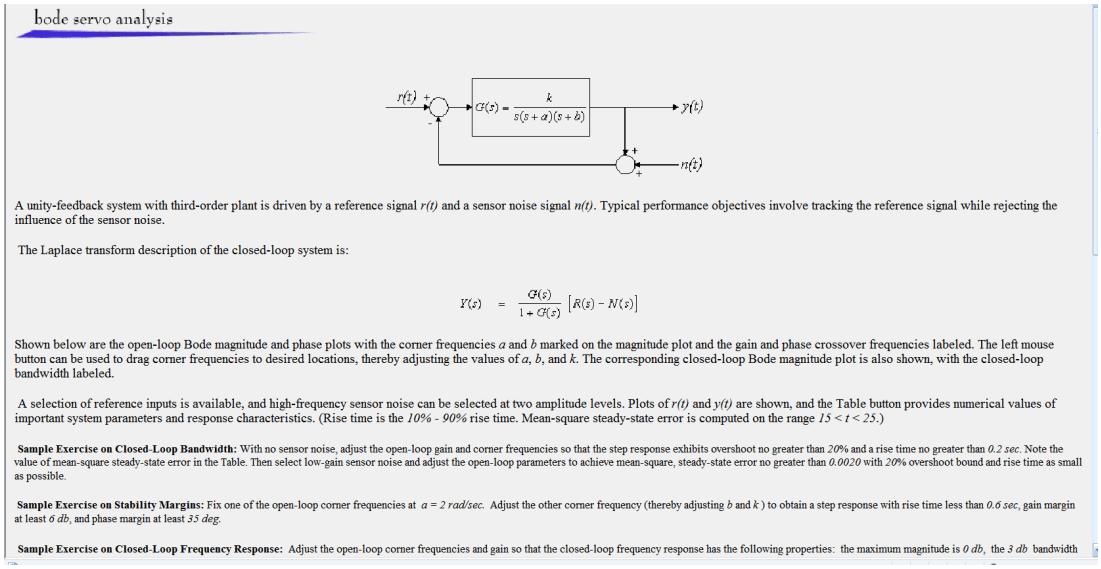
bode servo analysis
r(t)
+
-
G(s) = k / s(s + a)(s + b)
y(t)
+
+
n(t)
A unity-feedback system with third-order plant is driven by a reference signal r(t) and a sensor noise signal n(t). Typical performance objectives involve tracking the reference signal while rejecting the influence of the sensor noise.
The Laplace transform description of the closed-loop system is:
Y(s) = G(s) / (1 + G(s)) [R(s) - N(s)]
Shown below are the open-loop Bode magnitude and phase plots with the corner frequencies a and b marked on the magnitude plot and the gain and phase crossover frequencies labeled. The left mouse button can be used to drag corner frequencies to desired locations, thereby adjusting the values of a, b, and k. The corresponding closed-loop Bode magnitude plot is also shown, with the closed-loop bandwidth labeled.
A selection of reference inputs is available, and high-frequency sensor noise can be selected at two amplitude levels. Plots of r(t) and y(t) are shown, and the Table button provides numerical values of important system parameters and response characteristics. (Rise time is the 10% - 90% rise time. Mean-square steady-state error is computed on the range 15 < t < 25.)
Sample Exercise on Closed-Loop Bandwidth: With no sensor noise, adjust the open-loop gain and corner frequencies so that the step response exhibits overshoot no greater than 20% and a rise time no greater than 0.2 sec. Note the value of mean-square steady-state error in the Table. Then select low-gain sensor noise and adjust the open-loop parameters to achieve mean-square, steady-state error no greater than 0.0020 with 20% overshoot bound and rise time as small as possible.
Sample Exercise on Stability Margins: Fix one of the open-loop corner frequencies at a = 2 rad/sec. Adjust the other corner frequency (thereby adjusting b and k) to obtain a step response with rise time less than 0.6 sec, gain margin at least 6 db, and phase margin at least 35 deg.
Sample Exercise on Closed-Loop Frequency Response: Adjust the open-loop corner frequencies and gain so that the closed-loop frequency response has the following properties: the maximum magnitude is 0 db, the 3 db bandwidth

Figura 2.4 Vista en pantalla de Bode Servo Analysis

2.1.4 Sense and sensitivities [7]

Permite al usuario observar la respuesta al escalón al modificar la perturbación, modificar el gráfico de Bode de magnitud de ganancia y la ganancia al ruido en el lazo de retroalimentación. Además presenta la sensibilidad y la sensibilidad complementaria.

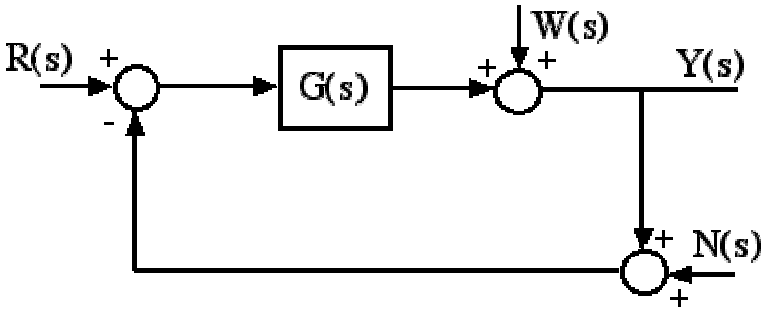

Figura 2.5 Modelo del sistema analizado en Sense and Sensitivities [7]

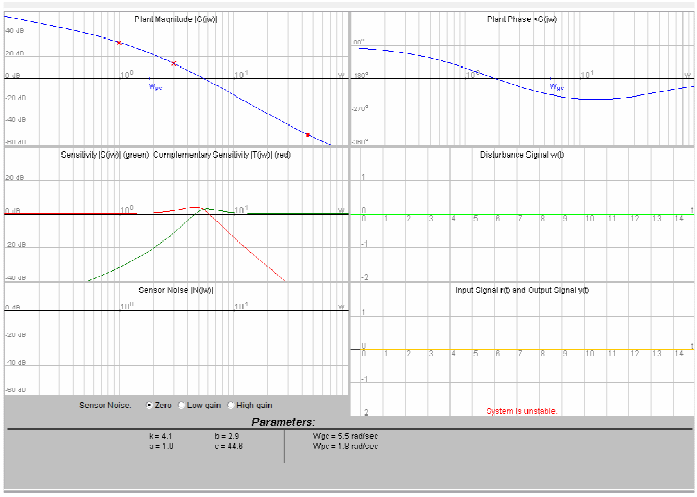

Figura 2.6 Vista en pantalla de Sense and Sensitivities

2.2 Sitio de Newcastle University (Australia) [8]

Este sitio presenta varios programas interactivos para observar algunas características de los sistemas de control. Está constituido por distintos módulos creados en Java, donde se presentan problemas reales de control.

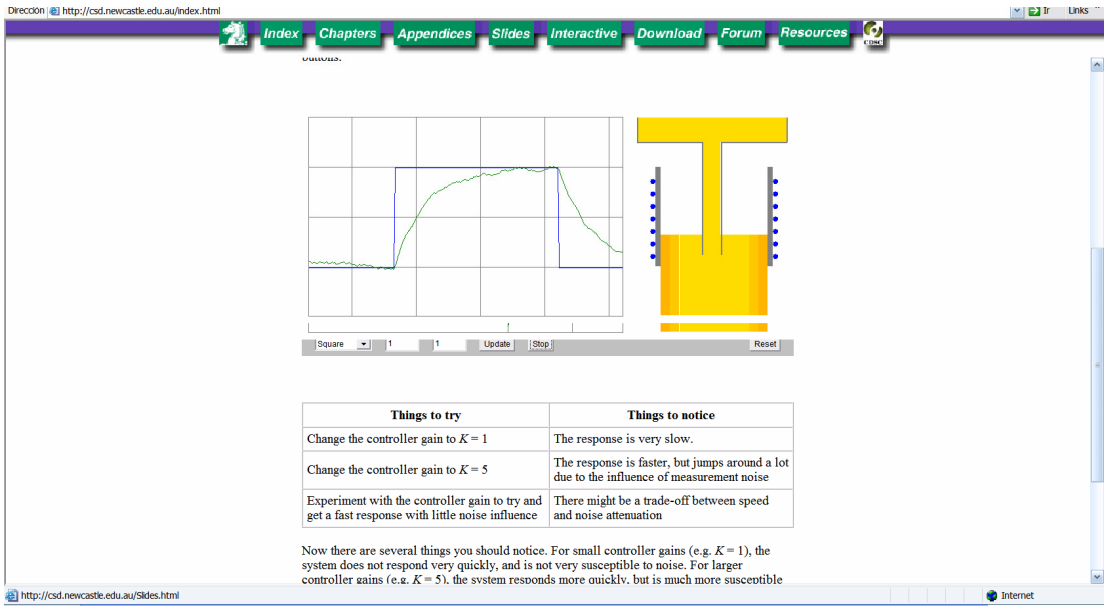
Dirección http://csd.newcastle.edu.au/index.html
Index
Chapters
Appendices
Slides
Interactive
Download
Forum
Resources
Square
Update
Stop
Reset
Things to try
Things to notice
Change the controller gain to K = 1
The response is very slow.
Change the controller gain to K = 5
The response is faster, but jumps around a lot due to the influence of measurement noise
Experiment with the controller gain to try and get a fast response with little noise influence
There might be a trade-off between speed and noise attenuation
Now there are several things you should notice. For small controller gains (e.g. K = 1), the system does not respond very quickly, and is not very susceptible to noise. For larger
http://csd.newcastle.edu.au/Slides.html
Internet

Figura 2.7 Vista en pantalla del sitio de la Universidad de Newcastle

Los casos que se presentan son los siguientes:

- Moldeado continuo
- Péndulo invertido
- Molino de rodillos
- Control de nivel de tanques
- Placa y bola
- Control de forma
- Columnas de destilación
- Control de pH

Algunos de los temas estudiados en estos módulos son:

- Lugar geométrico de las raíces
- Respuesta al escalón
- Sensibilidad al ruido
- Control en variables de estado
- Respuesta en frecuencia
- Control PI
- Tiempo muerto en la industria
- Observadores de estado

Este portal es sumamente atractivo y didáctico, ya que con cada problema de control se presenta su modelado matemático, con las explicaciones correspondientes así como fotografías de los sistemas estudiados. El hecho de que permita al usuario modificar los parámetros de dichos sistemas constituye un alto valor agregado para el tutorial.

Además tiene otra herramienta denominada "*Frequency Response Tool*" que permite al usuario insertar una función de transferencia cualquiera en su forma de polinomios de "s" y observar la respuesta al escalón y los diagramas de Bode de Magnitud y fase. Sumamente útil para el curso de control.

La desventaja de este sistema es que al estar programado en Java puede ser un poco lento en algunos equipos, debido a que la máquina virtual de Java ocupa mucha memoria.

2.3 Sitio de la Universidad de Exeter (Reino Unido) [9]

En este sitio encontramos la presentación de un problema de control de temperatura. El tutorial se compone de distintos pasos en los que se presenta una descripción del problema, una descripción de los componentes asociados al sistema. Luego se presentan los diferentes algoritmos de control y sus características (P, PI, PID y ON OFF). Se presentan además dos métodos de sintonización para el controlador PID.

Finalmente se presenta el módulo interactivo donde el usuario puede modificar los parámetros del sistema, los parámetros de la simulación y los parámetros del controlador (figura 2.8). Es posible observar la respuesta del sistema así como un diagrama de Bode cuando se selecciona el algoritmo de control PID.

Es un tutorial bastante práctico, sin embargo carece de explicaciones matemáticas que sustenten los resultados que se presentan. Aún así se presenta un problema interesante para ser estudiado por aquellos con conocimientos básicos de control.

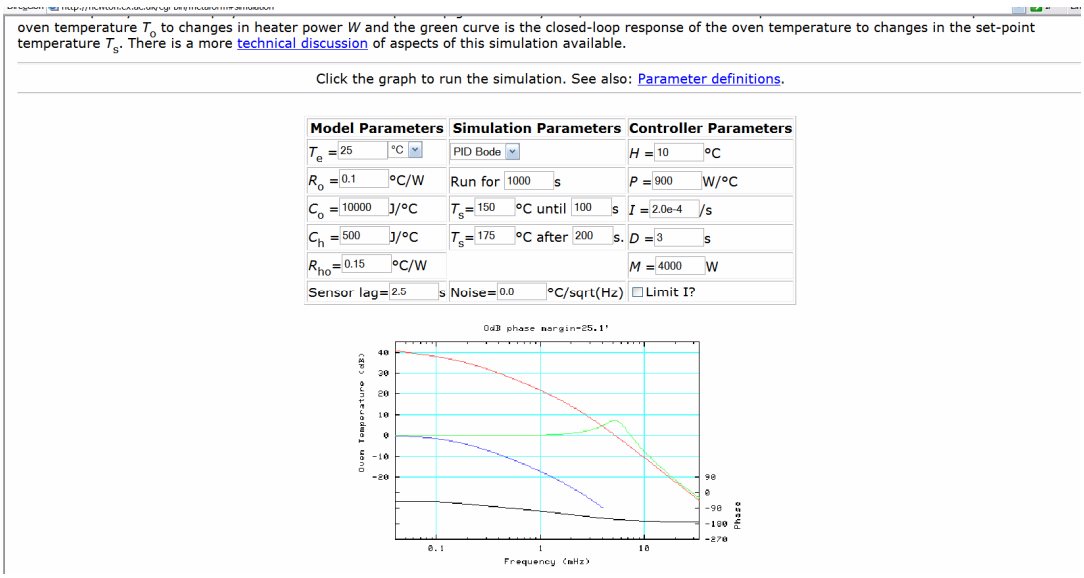
oven temperature T_o to changes in heater power W and the green curve is the closed-loop response of the oven temperature to changes in the set-point temperature T_s. There is a more technical discussion of aspects of this simulation available.
Click the graph to run the simulation. See also: Parameter definitions.
Model Parameters
Simulation Parameters
Controller Parameters
T_e = 25 °C
PID Bode
H = 10 °C
R_o = 0.1 °C/W
Run for 1000 s
P = 900 W/°C
C_o = 10000 J/°C
T_s = 150 °C until 100 s
I = 2.0e-4 /s
C_h = 500 J/°C
T_s = 175 °C after 200 s.
D = 3 s
R_{ho} = 0.15 °C/W
M = 4000 W
Sensor lag= 2.5 s
Noise= 0.0 °C/sqrt(Hz)
Limit I?
0dB phase margin=25.1°
Oven Temperature (dB)
Phase
Frequency (mHz)

Figura 2.8 Vista en pantalla del sitio de la Universidad de Exeter

2.4 Sitio de la Universidad de Tennesse en Chatannogga (E.U.A.) [10]

En el sitio de esta universidad es posible acceder vía Internet a un laboratorio real. Sin embargo para tener acceso a esta opción es necesario estar inscrito como usuario. Los experimentos y la adquisición de datos se dan en tiempo real, sin embargo si un mismo sistema físico quiere ser accedido por más de un usuario, este nuevo usuario entra en lista de espera debido a que el sistema al ser físico y real, solo puede ser utilizado por un usuario a la vez. Asimismo el usuario solo puede tener acceso a uno de los módulos que ofrece el laboratorio. Los sistemas que se incluyen son los siguientes:

- Voltaje
- Temperatura
- Caudal
- Nivel

- Velocidad
- Presión

Debido a la necesidad de inscripción previa para utilizar el equipo y que el propósito final de este proyecto no es la evaluación de los módulos de control en Internet no se presentan resultados de su utilización.

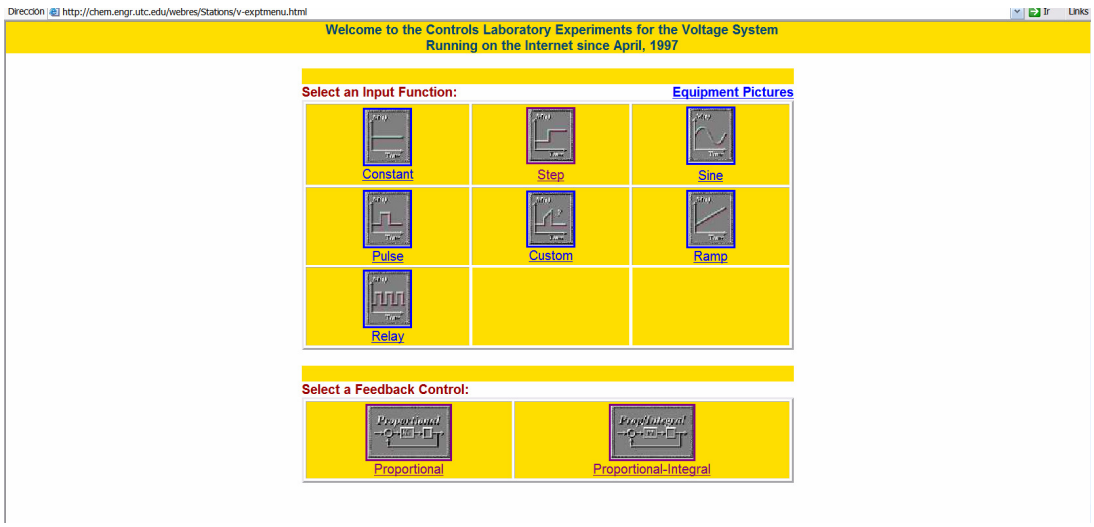

Figura 2.9 Sitio de la Universidad de Tennesse en Chatannogga

2.5 Sitio de la empresa TechTeach [11]

Desarrollado en Noruega por Finn Haugen para la empresa TechTeach, puede ser accedido de forma gratuita.

Este portal incluye una gran cantidad de simulaciones para los sistemas de control. Cada módulo de simulación viene con el archivo .exe correspondiente, la descripción del sistema a simular, los objetivos del módulo de simulación, así como una motivación para

ejecutar el módulo y las tareas a realizar en él. Para su ejecución es necesario contar con el programa LabView instalado o descargar el programa "LabVIEW Run-Time Engine" (LVRTE) el cual se consigue de forma gratuita, sin embargo el tamaño de este programa (32MB) lo hace poco atractivo para ser descargado por usuarios con conexión a Internet por medio de una conexión telefónica. Además no se le podría clasificar como un tutor por Internet ya que consta de programas que requieren instalación.

Figura 2.10 Vista en pantalla del sitio de TechTeach

2.6 Página personal de John Shaw [12]

Esta página personal presenta un tutorial no interactivo de sistemas de control PID así como la sintonización de los mismos. Entre los temas expuestos se encuentran:

- El lazo de control
- Las respuestas del proceso y los efectos sobre ella de las perturbaciones
- Algoritmos PID y sus características

Este tutorial puede servir como complemento para los estudiantes del curso de Control automático.

2.7 Sitio de la Universidad de Bochum en Ruhr (Alemania) [13]

En esta página se encuentra el VCLab, herramienta desarrollada con el propósito de simular experimentos de laboratorio vía Internet. Entre los problemas de control que se presentan encontramos:

- Bola y balanza
- Gyropéndulo
- 3 tanques

Este sitio presenta el inconveniente de que para su utilización se necesita contar con Matlab y Netscape así como descargar los complementos necesarios, lo que limita en gran medida su uso generalizado.

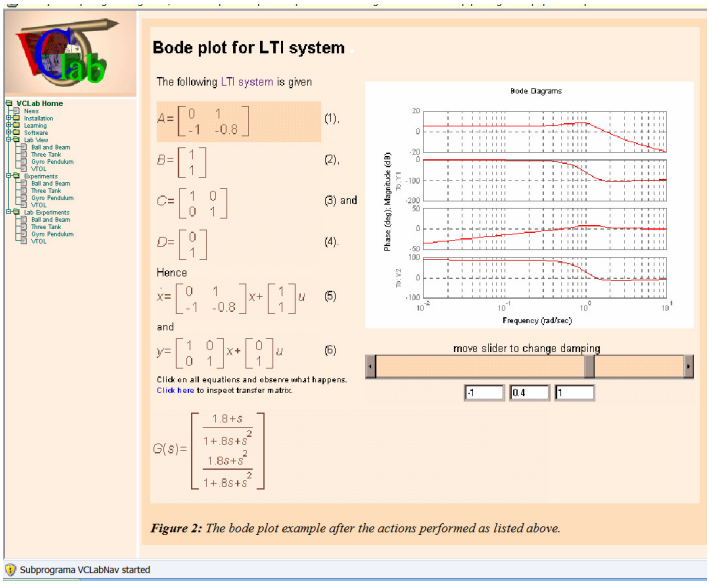

Figura 2.11 Vista en pantalla del sitio de la Universidad de Bochum

2.8 Sitio de la Escuela Superior de Ingenieros de Bilbao (España) [14]

El Departamento de Ingeniería de sistemas y automática de la Escuela Superior de Bilbao ofrece tutorías de controladores PID para Matlab. El tutorial consta de diez lecciones:

- Introducción al control
- Acción proporcional
- Acción integral
- Acción derivativa
- Introducción al PID

- Aspectos prácticos del PID
- El PID real
- Algoritmos y estructuras del PID
- Sintonía del PID
- Estrategias no lineales

Este tutorial presenta el inconveniente de que para su utilización se requiere tener instalado Matlab.

2.9 Softintegration [15]

En esta página se encuentra el lenguaje de programación Ch. Se encuentra también una interfaz donde se promociona la capacidad de este lenguaje para crear páginas Web interactivas orientadas al control automático.

En este sitio el usuario tiene la posibilidad de resolver problemas de sistemas continuos y discretos. Además, es posible describir estos sistemas en forma de variables de estado, polos y ceros o función de transferencia. Entre las funciones accesibles al usuario se encuentran:

- Análisis en el dominio del tiempo
- Análisis en el dominio de la frecuencia
- Diseño y análisis en variables de estado
- Lugar geométrico de las raíces

- Interconexión de sistemas lineales invariantes en el tiempo

Esta aplicación es sumamente rápida y versátil además de que no requiere instalar software adicional. Está programada en el mismo lenguaje que el proyecto desarrollado y es la que más se acerca al mismo.

CAPÍTULO 3: Lenguaje de programación Ch

3.1 Características generales

El lenguaje Ch es una implementación de C que incorpora características de C++, y otros lenguajes y paquetes, que permite crear listas de instrucciones ("*scripts*"), desarrollar aplicaciones rápidamente y su integración con varios sistemas. [1]

Es empotrable y puede utilizarse como una máquina de listas de instrucciones para aplicaciones y hardware de C/C++. Está especialmente diseñado para aplicaciones en ciencias e ingenierías. Cuenta con soporte gráfico incorporado, funciones matemáticas genéricas y arreglos computacionales para algebra lineal y cómputo matricial. Permite además hacer gráficos en dos y tres dimensiones, la solución de ecuaciones diferenciales, integración y análisis de Fourier entre otros.

Ch permite programar desde la consola utilizando un lenguaje de muy alto nivel.

Este lenguaje de programación comparte funciones con algunos otros lenguajes de programación y software.

- Al igual que C *shell* (consola de C), Ch se puede utilizar para programar desde la consola.
- Está diseñado y ha sido ampliamente utilizado para principiantes con experiencia computacional limitada al igual que Basic.
- Como Perl, permite la creación de Interfases de Pasarela en Modo Común (CGI por sus siglas en inglés).

- Al igual que Java se utiliza para cómputo en línea.
- Tal como lo hace JavaScript, los *scripts* en Ch se pueden empotrar en archivos HTML.
- Se puede utilizar para cómputo científico al igual que Fortran 77/90.

La figura 3.1, muestra la relación de Ch con los programas y lenguajes recién mencionados.

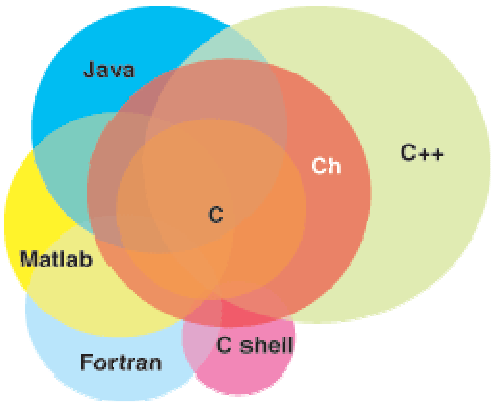

Figura 3.1 Relación de Ch con otros lenguajes de programación [1]

Algunas de las características más sobresalientes de Ch se muestran a continuación:

- Interpretativo
- Interactivo, los códigos de C se pueden escribir línea por línea
- Soporte gráfico y matemático poderoso
- Lenguaje de muy alto nivel, reduce la brecha entre lenguajes de alto y bajo nivel.

- Basado en objetos
- Puede ser utilizado en varias plataformas
- Seguro para trabajar en red
- Transportable
- Empotrable
- Listo para trabajar en la web

3.2 Capacidad para resolver problemas de control automático [3]

En la página www.softinteraction.com la cual provee Ch, se encuentra el "Ch Control System Toolkit". Esta herramienta está orientada exclusivamente a resolver problemas de control automático. Esta herramienta contiene una clase llamada Control cuyas funciones miembros se utilizan para el diseño, análisis y modelado de los sistemas de control.

Algunas de las características principales de esta herramienta son las siguientes:

- Compatible con C/C++
- Orientado a objetos
- Empotrable
- Soporta mútliples representaciones de sistemas discretos y continuos

- Conversión de discreto a continuo
- Interconexión de modelos
- Análisis en el dominio del tiempo
- Análisis en el dominio de la frecuencia
- Lugar geométrico de las raíces para sistemas SISO
- Controlabilidad y observabilidad
- Diseño en variables de estado
- Control óptimo y solución de ecuaciones
- Recursos gráficos

Las herramientas de sistemas de control de Ch realizan la gran mayoría de las tareas que realizan las herramientas equivalentes en Matlab.

3.3 Capacidad de desarrollo de sitios Web interactivos [2]

Ch cuenta con la capacidad de desarrollar Interfaz de Pasarela Común (CGI) por sus siglas en inglés. CGI es un estándar que especifica como programas en el exterior interactúan con un servidor Web. Una de las aplicaciones más importantes para el CGI es el manejo de formas interactivas en las que un programa en el servidor recibe argumentos por medio de las formas y genera una página Web en forma dinámica.

Una CGI puede ser escrita en cualquier lenguaje que le permita ejecutarse en el servidor. Ch por su parte incorpora varias características de seguridad para el manejo de este tipo de interfaces.

Para la implementación de CGI la biblioteca cgi.h incluye las clases CRequest, CResponse, CCookie y CServer cuyas funciones permiten interactuar con el usuario.

CAPÍTULO 4. Tutor interactivo para control automático

El tutor interactivo de control por Internet desarrollado consta de varios módulos, cada uno con una página Web donde se presenta una breve explicación acerca de su utilización y de la respuesta que se va a obtener. El portal principal incluye un menú donde se presentan los diferentes módulos. Durante la creación de los módulos se le brindó especial atención a la simplicidad, de tal manera que el usuario no tuviera que acceder a más de una página web para ingresar los datos del sistema a analizar.

La selección de los módulos se hizo comparando el programa del curso [16], exámenes de cursos lectivos anteriores [17] y las herramientas para el análisis de sistemas de control que se incluyen en el conjunto de herramientas de esta materia del lenguaje Ch [3]. De está manera se buscaba que el tutor incluyera los temas que son más comúnmente evaluados en los exámenes los cuales se enumeran en los apartados siguientes.

En cada módulo es posible acceder a un documento de ayuda que explica brevemente el funcionamiento del mismo, así como una breve explicación de la información requerida.

Una vez ingresados los datos por el usuario, estos son recibidos por la función de Ch correspondiente y son procesados para luego desplegar la respuesta en el navegador web.

Los módulos desarrollados incluyen los siguientes temas:

- Respuesta de sistemas de orden reducido
- Identificación de modelos

- Análisis de estabilidad
- Análisis de sistemas de control
- Sintonización de controladores PID

4.1 Respuesta de la planta a una entrada escalón

Se presenta la opción al usuario de ingresar plantas de primer orden, polo doble y segundo orden. Una vez que el usuario ingresa los datos de la planta, puede observar en pantalla el mapa de polos y ceros de la planta y su respuesta al escalón unitario. A continuación se presenta un breve ejemplo usando un modelo de segundo orden con ζ=0.7, ω_n=1 y kp=1.

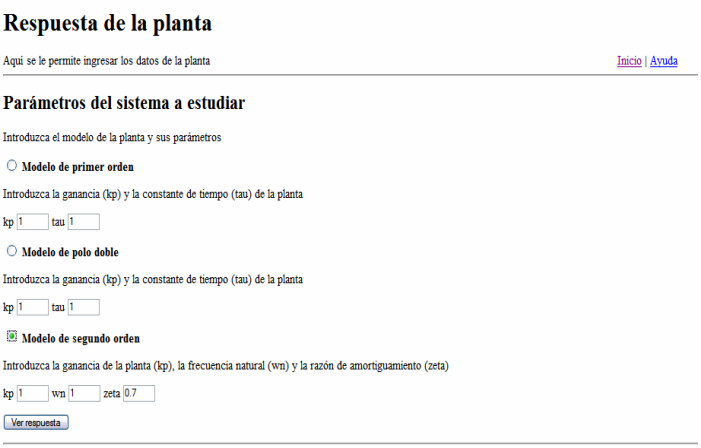
Respuesta de la planta
Aquí se le permite ingresar los datos de la planta
Inicio | Ayuda
Parámetros del sistema a estudiar
Introduzca el modelo de la planta y sus parámetros
Modelo de primer orden
Introduzca la ganancia (kp) y la constante de tiempo (tau) de la planta
kp 1 tau 1
Modelo de polo doble
Introduzca la ganancia (kp) y la constante de tiempo (tau) de la planta
kp 1 tau 1
Modelo de segundo orden
Introduzca la ganancia de la planta (kp), la frecuencia natural (wn) y la razón de amortiguamiento (zeta)
kp 1 wn 1 zeta 0.7
Ver respuesta

Figura 4.1 Pantalla de entrada de datos del módulo para respuesta de la planta

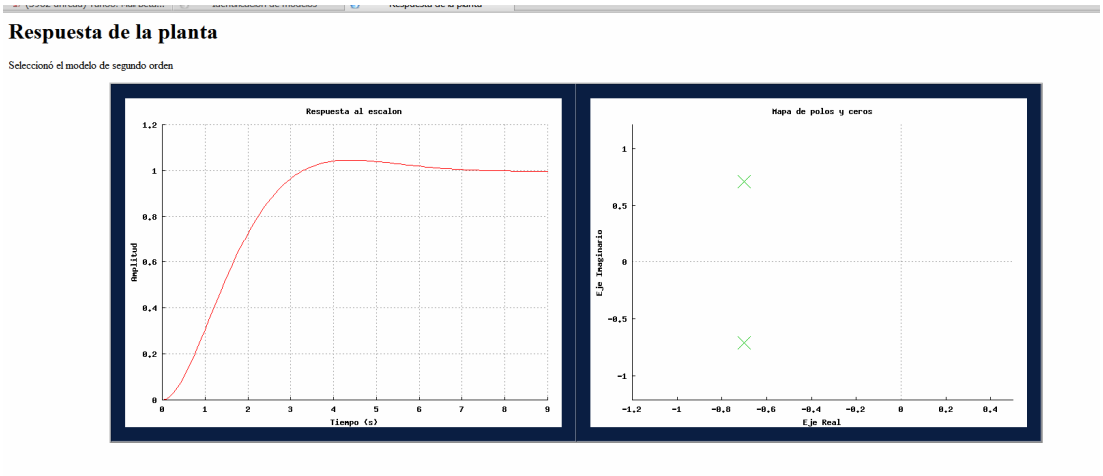

Figura 4.2 Curvas generadas por el módulo de respuesta de la planta

4.2 Identificación de modelos

En esta sección el usuario ingresa la función de transferencia del sistema en su forma de polos y ceros o como división de polinomios de la frecuencia compleja "s". Además, se le pide que ingrese el tipo de modelo a obtener ya sea este de polo simple más tiempo muerto, polo doble más tiempo muerto o segundo orden sobreamortiguado más tiempo muerto. Luego el programa genera la respuesta al escalón del sistema ingresado y utilizando los valores de esta se encarga de obtener el modelo solicitado. Seguidamente imprime en pantalla los parámetros del modelo obtenido y la respuesta al escalón del sistema original y su modelo para ser comparadas por el usuario. El módulo utiliza el método de identificación 123c de Alfaro[18]. Se muestra a continuación un ejemplo de la respuesta generada utilizando una planta de cuarto orden con la siguiente función de transferencia expresada en la forma de polos y ceros:

$$H(s) = \frac{1}{(s+0{,}625)(s+0{,}25)(s+0{,}5)(s+1)}$$

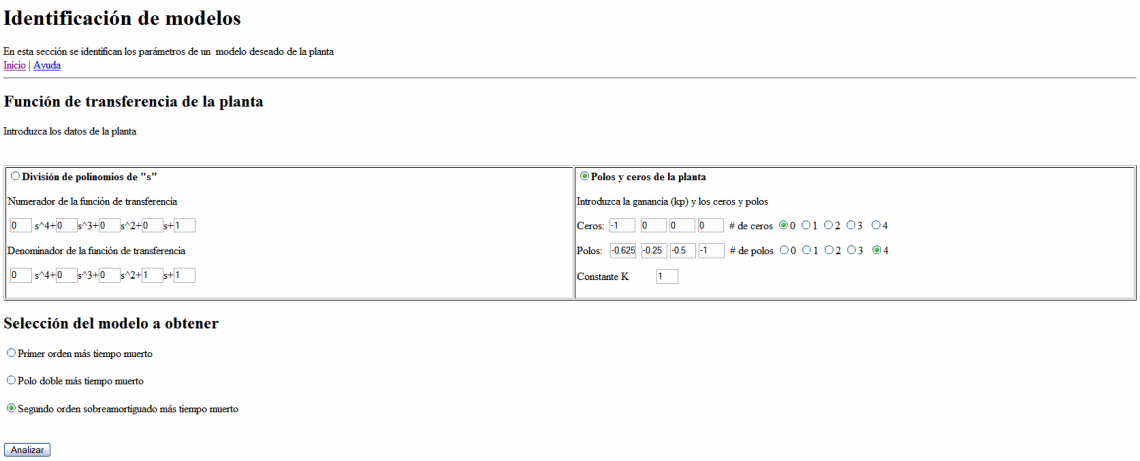
Identificación de modelos
En esta sección se identifican los parámetros de un modelo deseado de la planta
Inicio | Ayuda
Función de transferencia de la planta
Introduzca los datos de la planta
División de polinomios de "s"
Numerador de la función de transferencia
0 s^4+0 s^3+0 s^2+0 s+1
Denominador de la función de transferencia
0 s^4+0 s^3+0 s^2+1 s+1
Polos y ceros de la planta
Introduzca la ganancia (kp) y los ceros y polos
Ceros: -1 0 0 0 # de ceros 0 1 2 3 4
Polos: -0.625 -0.25 -0.5 -1 # de polos 0 1 2 3 4
Constante K 1
Selección del modelo a obtener
Primer orden más tiempo muerto
Polo doble más tiempo muerto
Segundo orden sobreamortiguado más tiempo muerto
Analizar

Figura 4.3 Pantalla de entrada del módulo para identificación de modelos

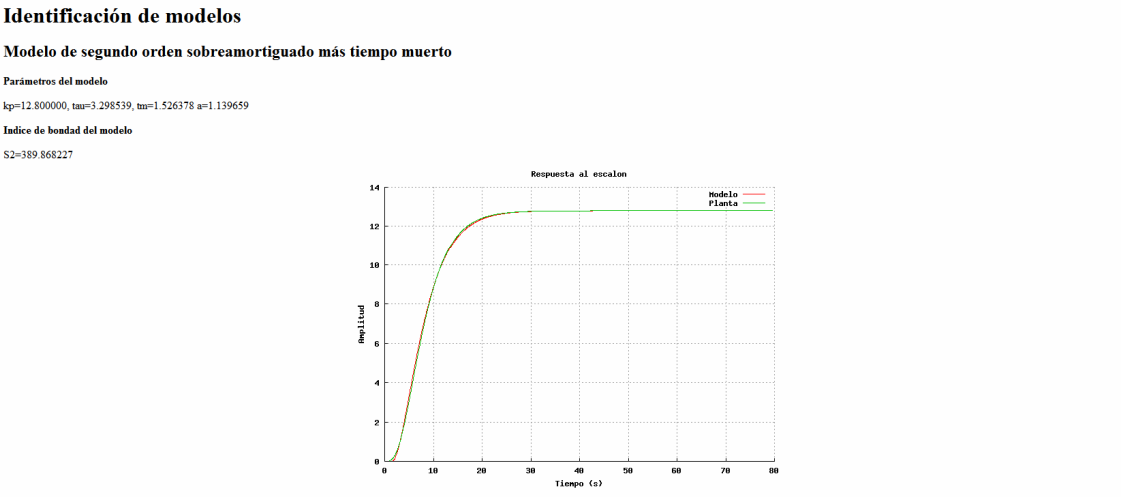

Figura 4.4 Respuesta del módulo de identificación de modelos

4.3 Herramientas clásicas para el análisis de sistemas de control

Aquí se le solicita al usuario ingresar la función de transferencia de lazo abierto de la planta *L(s)* como división de polinomios de la frecuencia compleja "s" o en su representación de polos y ceros. Seguidamente se selecciona entre una de las herramientas de análisis ya sea la gráfica del lugar geométrico de las raíces, el diagrama de Bode o el diagrama de Nyquist. Una vez que se ingresan los parámetros se presenta en pantalla la gráfica deseada.

Se presenta un ejemplo de la respuesta obtenida al solicitar la gráfica del lugar geométrico de las raíces para un sistema con la siguiente función de transferencia de lazo abierto:

$$L(s) = \frac{s+1}{s^2+1{,}5s+1}$$

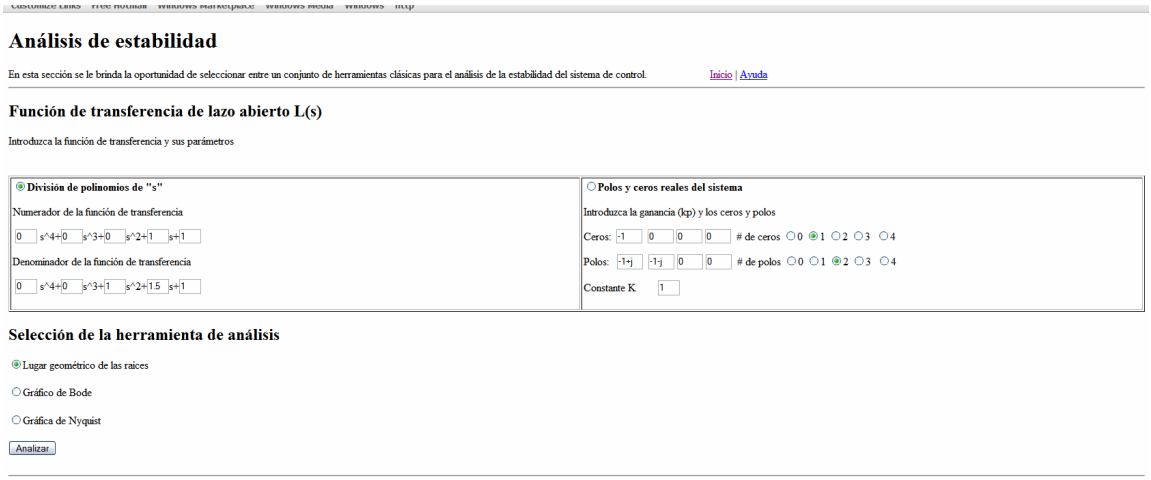
Análisis de estabilidad
En esta sección se le brinda la oportunidad de seleccionar entre un conjunto de herramientas clásicas para el análisis de la estabilidad del sistema de control.
Inicio | Ayuda
Función de transferencia de lazo abierto L(s)
Introduzca la función de transferencia y sus parámetros
División de polinomios de "s"
Numerador de la función de transferencia
0 s^4+0 s^3+0 s^2+1 s+1
Denominador de la función de transferencia
0 s^4+0 s^3+1 s^2+1.5 s+1
Polos y ceros reales del sistema
Introduzca la ganancia (kp) y los ceros y polos
Ceros: -1 0 0 0 # de ceros 0 1 2 3 4
Polos: -1+j -1-j 0 0 # de polos 0 0 1 2 3 4
Constante K 1
Selección de la herramienta de análisis
Lugar geométrico de las raíces
Gráfico de Bode
Gráfica de Nyquist
Analizar

Figura 4.5 Pantalla de entrada del módulo para análisis de estabilidad

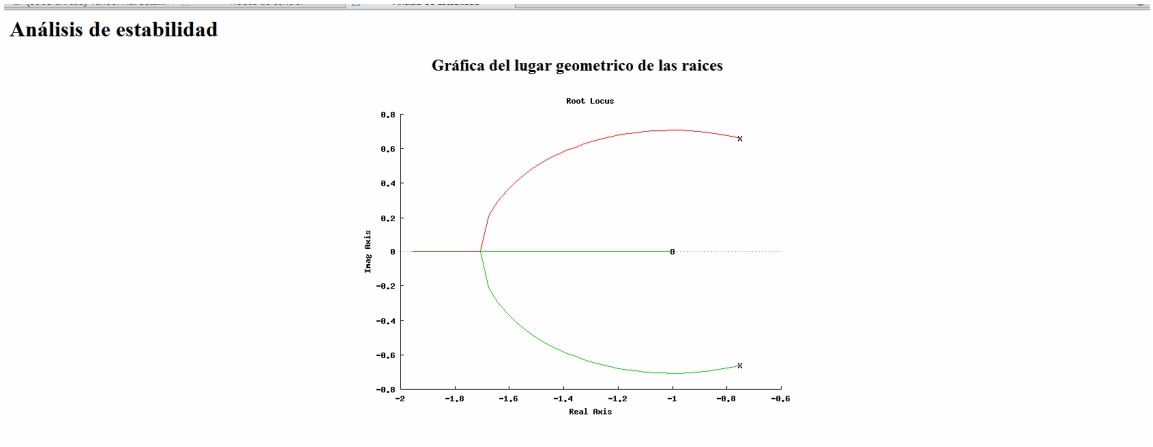

Figura 4.6 Respuesta del módulo para análisis de estabilidad

4.4 Controladores P, PI y PID

En esta sección se le pide al usuario ingresar la función de transferencia de la planta en cualquiera de sus dos formas y los parámetros del controlador a utilizar según el controlador seleccionado. Los controladores disponibles son el P, el PI, el PD y el controlador PID-Ideal. Una vez que el usuario ingresa esta información, el programa despliega en pantalla la respuesta a una entrada escalón en la perturbación contiguo a la respuesta a una entrada escalón en el valor deseado.

Se presenta a continuación el gráfico de la respuesta obtenida usando un controlador PI con un tiempo integral de 0,2s y una ganancia Kc=1. El modelo de la planta es el siguiente:

$$G_p(s) = \frac{1}{s+1}$$

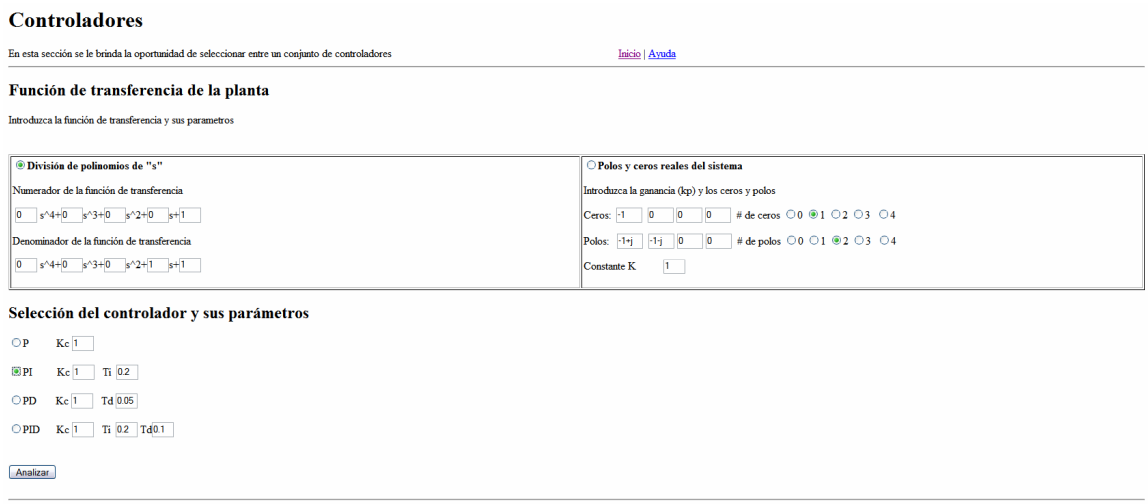
Controladores
En esta sección se le brinda la oportunidad de seleccionar entre un conjunto de controladores
Inicio | Ayuda
Función de transferencia de la planta
Introduzca la función de transferencia y sus parametros
División de polinomios de "s"
Numerador de la función de transferencia
0 s^4+0 s^3+0 s^2+0 s+1
Denominador de la función de transferencia
0 s^4+0 s^3+0 s^2+1 s+1
Polos y ceros reales del sistema
Introduzca la ganancia (kp) y los ceros y polos
Ceros: -1 0 0 0 # de ceros 0 1 2 3 4
Polos: -1+j -1-j 0 0 # de polos 0 1 2 3 4
Constante K 1
Selección del controlador y sus parámetros
P Kc 1
PI Kc 1 Ti 0.2
PD Kc 1 Td 0.05
PID Kc 1 Ti 0.2 Td 0.1
Analizar

Figura 4.7 Pantalla de entrada del módulo de controladores

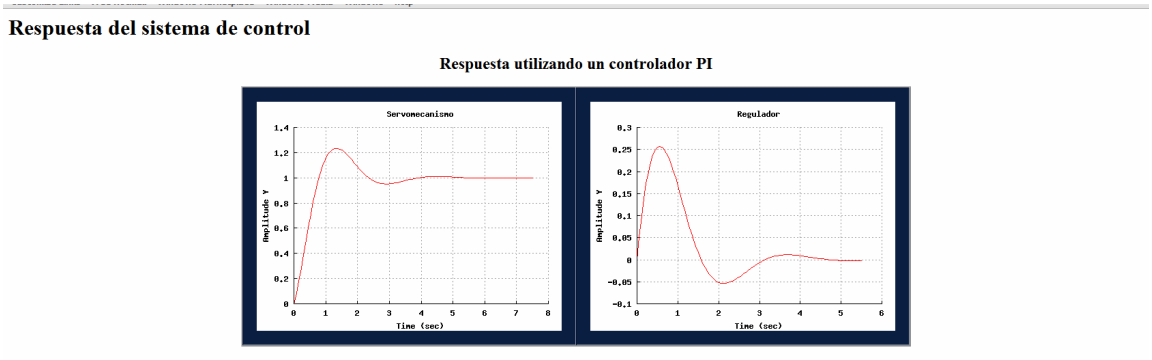

Figura 4.8 Respuesta del módulo para modos de control

4.5 Sintonización de controladores

El módulo de sintonización requiere que el usuario indique el tipo de controlador que quiere obtener y que ingrese la información de la planta a controlar. La función de transferencia se puede expresar en su forma de división de polinomios de frecuencia compleja "s" o en su forma de polos y ceros. Los controladores disponibles son el P, PI y PID.

El usuario puede además escoger entre varios métodos de sintonización. Los métodos disponibles son:

- Zieglers y Nichols
- López
- Rovira
- Arrieta para regulador
- Arrieta para servomecanismo
- Solera para regulador
- Solera para servomecanismo

Para cada uno de los métodos el programa calcula el modelo requerido por el método de sintonización ya sea de polo simple o polo doble. Una vez que se calcula el modelo, se obtienen los parámetros correspondientes al controlador deseado. En aquellos casos que el método este definido para varios criterios de desempeño, estos son tomados en cuenta. Finalmente se despliegan en pantalla las respuestas correspondientes a una entrada escalón en el valor deseado y una entrada escalón en la perturbación a los lazos de control.

Se presenta un ejemplo para un controlador PID utilizando el método de Solera como regulador para una planta con la siguiente función de transferencia:

$$G_p(s) = \frac{1}{8s^3 + 12s^2 + 6s + 1}$$

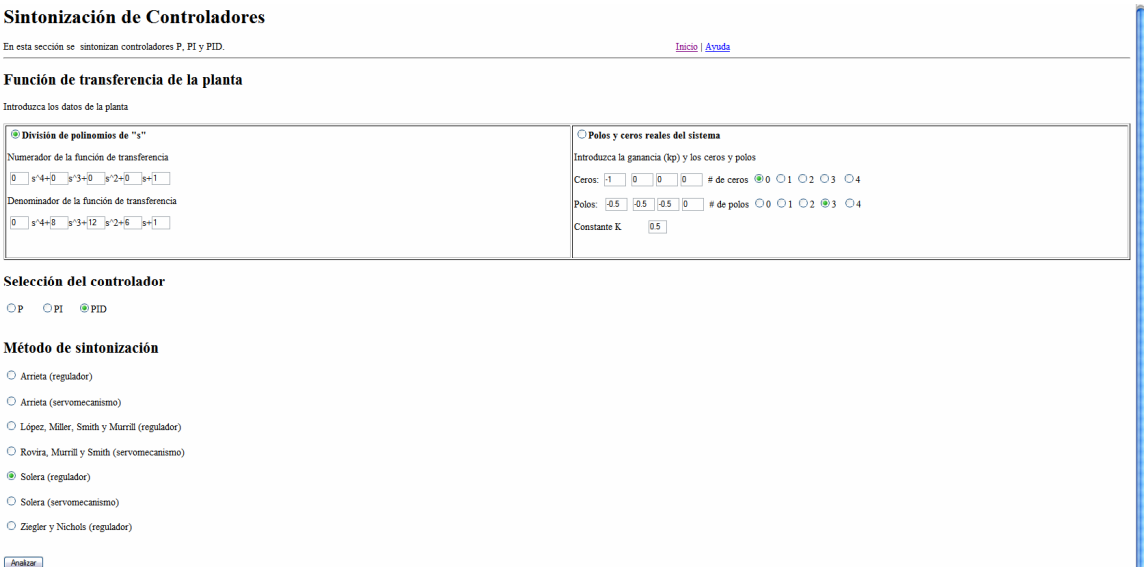
Sintonización de Controladores
En esta sección se sintonizan controladores P, PI y PID.
Inicio | Ayuda
Función de transferencia de la planta
Introduzca los datos de la planta
División de polinomios de "s"
Numerador de la función de transferencia
0 s^4+0 s^3+0 s^2+0 s+1
Denominador de la función de transferencia
0 s^4+8 s^3+12 s^2+6 s+1
Polos y ceros reales del sistema
Introduzca la ganancia (kp) y los ceros y polos
Ceros: -1 0 0 0 # de ceros 0 1 2 3 4
Polos: -0.5 -0.5 -0.5 0 # de polos 0 1 2 3 4
Constante K 0.5
Selección del controlador
P PI PID
Método de sintonización
Arrieta (regulador)
Arrieta (servomecanismo)
López, Miller, Smith y Murrill (regulador)
Rovira, Murrill y Smith (servomecanismo)
Solera (regulador)
Solera (servomecanismo)
Ziegler y Nichols (regulador)
Analizar

Figura 4.9 Pantalla de entrada para el módulo de sintonización de controladores

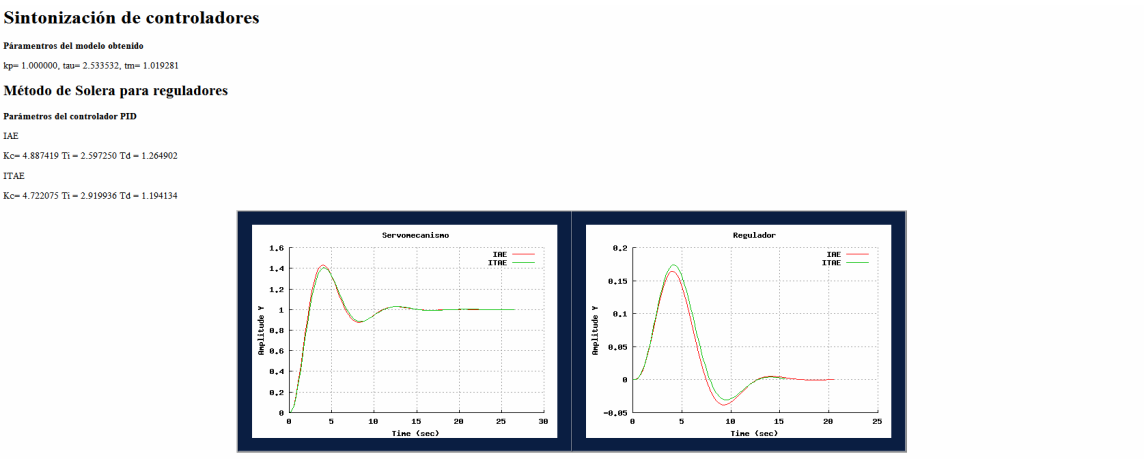

Figura 4.10 Respuesta del módulo de sintonización

CAPÍTULO 5: Conclusiones y recomendaciones

A continuación se listan las principales conclusiones derivadas de la elaboración del software y de la investigación realizada previamente.

- En Internet hay disponible cantidad importante de información y tutoriales relacionados al tema de control sin embargo no hay ninguno que se adapte a los contenidos del curso IE-431Sistemas de Control
- Muchos de las herramientas disponibles requieren la descarga de programas lo que resta flexibilidad a su utilización
- La mayoría están orientados al análisis de problemas específicos y no permiten al usuario ingresar los datos de su problema particular
- Usualmente se utiliza la respuesta al escalón y la gráfica de Bode como herramientas de análisis
- Ch es un lenguaje con un nivel de complejidad adecuado para aquellos con conocimientos básicos de C
- La herramienta para la creación de CGI implementada en Ch es simple y facilita la creación de este tipo de interfaces al estar orientada a objetos
- Se lograron desarrollar los módulos propuestos de manera satisfactoria
- Ch se puede utilizar para crear herramientas interactivas para otros cursos como análisis de sistemas o cursos de variable discreta.

De la puesta en marcha del proyecto se derivan las siguientes recomendaciones con miras al mantenimiento y la extensión del tutor desarrollado.

- Para mantener y desarrollar la plataforma hay que tener como mínimo conocimientos básicos de programación y configuración de servidores web así como conocimiento de HTML para la creación de las páginas.
- Se recomienda mantener la independencia entre los módulos para facilitar el mantenimiento y extensión de la aplicación
- Es posible aumentar el orden de los sistemas a analizar sin embargo esto puede resultar en una pérdida de rapidez o simplicidad de la interfaz con el usuario
- No existe mucha documentación de aplicaciones desarrolladas en Ch sin embargo, el soporte técnico disponible para el grupo de usuarios de Ch en Yahoo! es bastante eficiente resolviendo dudas y es accesible de manera gratuita
- Para facilitar la comprensión del código fuente se elaboró una guía para el programador que contiene las principales funciones utilizadas.

Entre los temas que se pueden incluir para aumentar la utilidad de la aplicación se encuentran:

- Respuesta para plantas de orden superior
- Diseño de redes de compensación
- Determinación de parámetros de estabilidad tales como sensibilidad máxima
- Representación de sistemas en el espacio de estados
- Transformación de modelos
- Reducción de modelos
- Análisis de controlabilidad

BIBLIOGRAFÍA

[1] **"Ch—an embeddable C/C++ interpreter"**, http://www.softintegration.com/

[2] **"The Ch Language Environment CGI Toolkit",** http://www.softintegration.com/

[3] **"Ch Control System Toolkit User's Guide",** http://www.softintegration.com/

[4] **"LTI Arcade",** http://www.jhu.edu/~signals/arcade2/indexarcade2.htm

[5] **"Exploring the s-plane",** http://www.jhu.edu/~signals/explore/index.html

[6] **"Bode Servo Analysis",** http://www.jhu.edu/~signals/bandwidth/index.html

[7] **"Sense and sensitivites",** http://www.jhu.edu/~signals/sensitivity/index.htm

[8] Goodwin, Graebe & Salgado. **"Control System Design"**, http://csd.newcastle.edu.au/

[9] "Temperature Controller Simulation"

http://newton.ex.ac.uk/teaching/CDHW/Feedback/OvSimForm-gen.html

[10] "**Resource Center for Engineering Laboratories on the Web",**

http://chem.engr.utc.edu/

[11] **"KYBSIM"** http://techteach.no/kybsim/index_eng.htm

[12] Shaw, J. **"PID Tutotial",** http://www.jashaw.com/pid/tutorial/

[13] "**Virtual Control Lab",** http://www.esr.ruhr-uni-bochum.de/VCLab/

[14] **"Tutorial sobre controladores PID",**

http://disa.bi.ehu.es/spanish/tutoriales/pid/leeme.html

[15] **"Web-Based Control System Design and Análisis"**,

http://www.softintegration.com/webservices/control/

[16] Descripción del curso IE-0423. Sistemas de Control. Escuela de Ingenierìa Eléctrica. Universidad de Costa Rica, II-2006.

[17] Exámenes del curso IE-0423. Sistemas de Control. Escuela de Ingenierìa Eléctrica. Universidad de Costa Rica, 2002-2005.

[18] Alfaro, V. Método de identificación de tres puntos 123c. Escuela de Ingeniería Eléctrica. Universidad de Costa Rica, agosto 2006.

[19]Alfaro, V. Métodos de sintonización. Escuela de Ingeniería Eléctrica. Universidad de Costa Rica. Junio 2006.

APÉNDICES

Guía del Programador

Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

IE – 0502 Proyecto Eléctrico

Tutor de control interactivo para Internet desarrollado en Ch

Guía del Programador

Por:

Alejandro Mora Sojo

Ciudad Universitaria Rodrigo Facio
Diciembre del 2006

Instrucciones de instalación

El programa Ch está disponible en www.softintegration.com y ahí mismo se encuentra la información necesaria para la instalación del software.

Instrucciones de configuración

La versión instalada de Ch en el servidor es Ch 5.5.0 y se ejecuta de manera adecuada, sin embargo se recomienda revisar el sitio www.softintegration.com en caso de que se le hayan hecho adiciones a las herramientas de control.

Es necesario contar con un servidor web. El servidor actualmente utilizado es arenal.eie.ucr.ac.cr el cual cuenta con Apache 2 y cuenta con el sistema operativo Linux.
Tanto los documentos html como los programas para el despliegue de los gráficos y manejo de información se encuentran en la misma carpeta. La carpeta es /var/www/ch.

Para poder ejecutar los *CGI* desde esta carpeta es necesario entrar a la configuración de Apache y cambiar el permiso correspondiente. Asimismo es necesario agregar .ch como un manejador de *CGI*. Para más información acerca de la configuración de Apache se puede ingresar a www.apache.org.
Es necesario darle permisos de lectura y ejecución a los archivos que se encuentren en /var/www/ch.

Formato general de los módulos

Cada módulo esta debidamente comentado para poder obtener una comprensión de lo que hace y cómo lo hace. A continuación se da una explicación del formato general de cada módulo y se brindan detalles de cómo se implementa cada sección.

Composición de los módulos

Los módulos que incluyen una o más respuestas gráficas constan de una página html, un programa en Ch, al cual se llamará "modulo1.ch" para efectos de esta explicación, que recibe los parámetros y traspasa los que corresponden a la o las funciones encargadas de generar los gráficos. A estos programas que generan los gráficos se les denominará para efectos de esta guía modulo1_image.ch

Estructura básica de módulo1.ch

Primero hay que tener en cuenta que el propósito de este programa es desplegar en pantalla la respuesta obtenida en un documento html. También se encargará de llamar a las funciones que generan las imágenes y pasarles los parámetros correspondientes.

A continuación se estudiará cómo se componen estos programas utilizando como ejemplo partes del programa modulo3_1.ch

-Sección de encabezado

```
#!/bin/ch
#include <cgi.h>
```

En esta sección se determina la localización del intérprete ch y se incluyen las librerías a utilizar. La librería cgi.h es la que permite obtener la información insertada en la página Web.

-Declaración de variables y clases

```
int main() {
  int  i, k;
  chstrarray name[13],values[13];      /*declaración de variables y clases*/
  char *modelo, *tool;
  class CResponse Response;
  class CRequest Request;
  class CServer Server;
```

En esta sección se declaran las variables que se utilizarán así como los objetos correspondientes a las clases definidas en cgi.h que permitirán obtener la información insertada por el usuario.

Class CResponse Request: define un objeto Request de clase CRequest cuyas funciones miembro nos permitirán obtener los parámetros introducidos por el usuario en la página Web.

class CResponse Response: define un objeto Response de clase CResponse cuyas funciones miembro nos permitirán crear la página web que el usuario verá como respuesta.

class CServer Server: define un objeto Server de clase CServer que nos permitirá llamar a las funciones encargadas de generar los gráficos como si se llamaran desde un servidor Web.

-Creación del tipo de respuesta

```
Response.setContentType("text/html");

 Response.begin();
```

*Response.title("Análisis de estabilidad"); /*titulo de la respuesta*/*
*setbuf(stdout,NULL); /*limpia el buffer*/*
*printf("<H1>Análisis de estabilidad</H1>\n"); /*Imprime el titulo de la respuesta*/*

Response.setContentType("text/html"): establece que el tipo de respuesta a genera res un archive html.

Response.begin(): inicia la respuesta.

Response.title("An´lisis de estabilidad"): establece el título de la respuesta que se genera.

printf("<H1>Análisis de estabilidad</H1>\n"): Imprime en el documento html que se generá como respuestas. Notese que es necesario insertar las etiquetas html correspondientes para lograr el formato deseado.

-Obtención de parámetros y asignación a variables

*modelo=Request.getForm("modelo"); /*obtención del parámetro modelo*/*

values[12]=Request.getForm("np");

modelo=Request.getForm("modelo"): Asigna a la variable tipo char apuntada por modelo el contenido del valor del parámetro "modelo" enviado por el usuario.

values[12]=Request.getForm("np"): Asigna a values[12] el valor del parámetro "np" enviado por el usuario.

-Despliegue de información en pantalla

printf(" <H2>Gráfica del lugar geometrico de las raices</H2> </p>");

Para imprimir en el documento de texto/html generado se utiliza la función printf y se incluyen las etiquetas html correspondientes al formato deseado. De igual manera se pueden crear tablas o hipervínculos.

-Llamado de la función que grafica

Este llamado se debe de realizar entre las funciones printf que establecen la localización de cómo queremos mostrar la figura según se muestra a continuación.

```
printf("<center><img src=\"/cgi-bin/ch/modulo3_image_1.ch");        for(i=0;i<k;i++){
                putc(i==0 ? '?' : '&', stdout);
                fputs(Server.URLEncode(name[i]),stdout);
                putc('=',stdout);
                fputs(Server.URLEncode(values[i]),stdout);
        }
    printf("\">\n</center>\n");
```

Esto equivaldría escribir:

printf(" <img src=\"/cgibin/ch/modulo3_image_1.ch?name[0]=values[0]&name [1]=values[1]&....&name[k-1]=values[k-1]\">\n \n");

Nótese que esto es equivalente a llamar modulo3_image_1.ch usando el método *GET* para *CGI.* De manera que los valores que se le envíen correspondan a parámetros con los nombres en el vector name y valores en el vector values.

También nótese que se desea que la figura esté centrada por lo tanto se debe llamar a la función en medio de las etiquetas html correspondientes.

-Fin de la respuesta

```
Response.end();
```

Estructura de modulo1_image.ch

-Sección de encabezado

```
#!/bin/ch
#include <cgi.h>
#include <control.h>
#include <chplot.h>
```

En esta sección se determina la localización del intérprete ch y se incluyen las librerías a utilizar. La librería cgi.h es la que permite obtener la información insertada en la página Web. Control.h contiene las funciones correspondientes a las herramientas de control automático y chplot.h al manejo de gráficos.

-Declaración de variables y clases

```
int main() {

  int k, nz, np;              /*declaración de variables y clases*/
  chstrarray name,value;

  class CResponse Response;
  class CRequest Request;
  class CPlot plot;
  class CControl sys;

  double den[5];
  double num[5];
  double kp;
  char *modelo, *tool;
```

class CPlot plot: crea un objeto plot perteneciente a la clase CPlot.
Class Control sys: crea un objeto sys perteneciente a la clase Control

-Obtención de parámetros y asignación a variables

*Request.getFormNameValue(name,value); /*asignación a name y value de los nombres y valores de los parámetros recibidos*/*

Request.getFormNameValue(name,value): asigna a los vectores name y values el nombre y valor de los parámetros enviados por modulo1.ch en el orden en que se enviaron.

num[k]=atof(value[k]);

Esta línea asigna a num[k] el valor de value[k]. La función atof obtiene la representación en double de value [k] el cual es de tipo char.

-Creación del modelo del sistema

```
sys.model("tf", num, den);
sys.model("zpk", zeros, polos, kp);
```

Estas funciones se usan en los casos en que contemos con el vector de coeficientes del numerador y el denominador de la función de transferencia y en el caso de que tengamos el vector de polos y ceros y su ganancia respectivamente. De esta manera se define el sistema sys.

-Creación de respuesta, graficación y finalización

```
Response.setContentType ("image/png"); /*creación de un respuesta tipo imagen*/
Response.begin();
sys.margin(&plot,NULL,NULL,NULL,NULL);
plot.outputType(PLOT_OUTPUTTYPE_STREAM, "png color");
plot.plotting();
Response.end();
```

Response.setContentType(“image/png”): establece que el formato de salida será una imagen tipo png.

Response.begin():inicia la respuesta.

sys.margin(&plot,NULL,NULL,NULL,NULL): crea un gráfico en plot.

plot.outputType(PLOT_OUTPUTTYPE_STREAM, "png color"): establece el formato de salida de la imagen de forma que pueda ser interpretada por el navegador web.

plot.plotting(): refresca la imagen con las características deseadas.

Response.end(): Finaliza la imagen

Resumen de funciones más utilizadas

a) Funciones del conjunto de herramientas para sistemas de control

Class CControl sys: crea un objeto sys de la clase Control

CControl::model: crea un modelo para un sistema lineal e invariante con el tiempo

Prototipo:

int model(string_t tipo,…/* [array double num[&], array double den[&], [double ts]] , [array double complex z[&], array double complex p[&], double k, [double ts]]

Parametros

tipo: string que indica el modelo a obtener. Se usa "tf" y "zpk" para división de polinomios de la frecuencia compleja s y para la forma de polos y ceros respectivamente

num: vector con los coeficientes del numerador de la función de transferencia

den: vector con los coeficientes del denominador de la función de transferencia

z: vector complejo con los ceros de la función de transferencia

p: vector complejo con los polos de la función de transferencia

k:constante k de ganancia

ts:tiempo de muestreo (para sistemas discretos)

Valor de retorno

Cero si el sistema se creo adecuadamente

CControl::step: calcula y grafica la respuesta a una entrada escalón unitario

Prototipo:

int step(class CPlot *plot, array double &yout, array double &tout, array double &xout,…/*tf*/)

Parámetros:

Plot: puntero a un objeto de la clase CPlot

Yout: vector de referencia que contendrá la amplitud de la respuesta

Tout: vector de referencia que tendrá el vector de tiempo utilizado para la simulación

Xout: vector con las trayectorias de los estados

Tf: tiempo final de la simulación

Valor de retorno

Cero si la operación es exitosa

CControl::margin: calcula y grafica de Bode y calcula los márgenes de ganancia y fase

Prototipo:

int step(class CPlot *plot, double *gm, double *pm, double *wcg, double *wcp, ……/*array double mag[&], array double phase[&], array double w[&]*/)

Parámetros:

Plot: puntero a un objeto de la clase CPlot

gm:puntero a variable que almacenará el margen de ganancia

pm:puntero a variable que almacenará el margen de fase

wcg: puntero a varible que tendrá la frecuencia de cruce por 0 dB

wcp: puntero a variable que tendrá la frecuencia de cruce por -180 grados

mag: vector de referencia que contendrá la amplitud de la respuesta en frecuencia

phase: vector de referencia que contendrá la fase de la respuesta en frecuencia

w: vector que tendrá las frecuencias utilizadas para la simulación

Tf: tiempo final de la simulación

Valor de retorno

Cero si la operación es exitosa

CControl::feedback: crea un sistema a partir de la realimentación

Prototipo:

CControl *feedback(CControl *sys2,…./¨*array int feedin[:], array int feedout[:], int sign */)

Parámetros:

sys2: puntero a un objeto de la clase CControl ubicado en el lazo de retroalimentación

feedin: vector que contiene los índices de las entradas involucradas en el lazo de retroalimentación

feedout: vector que contiene los índices las salidas involucradas en el lazo de retroalimentación

sign: signo de la retroalimentación, puede ser 1 o -1

Valor de retorno

Puntero no nulo al objeto CControl que representa el lazo de control

CControl::series: crea un sistema a partir de la conexión en serie de dos sistemas

Prototipo:

CControl *series(CControl *sys2,…./¨*array int output1[:], array int input2[:]*/)

Parámetros:

sys2: puntero a un objeto de la clase CControl ubicado en el lazo de retroalimentación

output1: vector que contiene los índices de las salidas a conectar

input2: vector que contiene los índices de las entradas de sys2 que se van a conectar

Valor de retorno:

Puntero no nulo al objeto CControl que representa la conexión en serie

CControl::nyquist: calcula y grafica el diagrama de Nyquist

Prototipo:

int nyquist(CPlot *plot, array double re[&], array double im[&], array double wout[&]…./¨*array double w[&] ó double wmin, double wmax*/)

Parámetros:

plot: puntero a un objeto de la clase CPlot donde se grafica la respuesta

re: vector de referencia que contendrá la parte real de la respuesta en frecuencia

im: vector de referencia que contendrá la parte imaginaria de la respuesta en frecuencia

wout: vector de referencia que contendrá las frecuencias utilizadas en rad/seg

w: vector de frecuencia a utilizar para el cálculo

wmin, wmax: límite inferior y superior del intervalo de frecuencia a utilizar

input2: vector que contiene los índices de las entradas de sys2 que se van a conectar

Valor de retorno

Cero si la operación fue exitosa

CControl::rlocus: calcula y grafica la gráfica del lugar geométrico de las raíces

Prototipo:

int rlocus(CPlot *plot, array double complex &r, array double &kout,…./¨*array double &k*/)

Parámetros:

plot: puntero a un objeto de la clase CPlot donde se grafica la respuesta

r: vector de referencia con la localización de las raíces

kout: vector de referencia que contendrá las ganancias utilizadas para la simulación

k:vector de referencia con las ganancias específicadas por el usuario

Valor de retorno

Cero si la operación fue exitosa

CControl::grid: genera grilla para los gráficos

Prototipo:

int grid (int flag)

Parámetros:

flag: Entero usado para activar o desactivar la grilla. 1 la activa y 0 la desactiva

Valor de retorno:

Cero si la operación fue exitosa

b) Funciones del conjunto de herramientas para *CGI*

Class CResponse Response: crea objeto Response de la clase CResponse. Se utiliza para generar respuestas en un CGI

CResponse::setContentType: define el tipo de respuesta

Prototipo:

Int setContentType(chchar * type);

Parámetros:

type: string que contiene el tipo valor del tipo de respuesta a obtener. Admite text/plain,

text/html, image/png, etc.

Valor de retorno:

Cero si se realiza la operación de manera exitosa

CResponse::begin: inicia el envío de contenido

Prototipo:

int begin();

Parámetros:

Ninguno

Valor de retorno:

0 si se completa exitosamente
CResponse::title

CResponse::end: detiene la salida estándar

Prototipo:

void end();

Parámetros:

Ninguno

Valor de retorno:

Ninguno

CResponse::title: define el título de la página HTML

Prototipo:

Void title(chchar *title);

Parámetros:

title: string que contiene el título deseado

Valor de retorno:

Ninguno

Class CRequest Request: crea objeto Request de la clase CRequest. Se utiliza para obtener los parámetros de un CGI

CRequest::getForm: retorna el valor correspondiente a un nombre que haya sido recibido vía POST o GET

Prototipo:

chchar *getForm(chchar *name);

Parámetros:

name: nombre del ítem deseado

Valor de retorno:

El valor correspondiente al ítem deseado

CRequest::getFormNameValue: obtiene todos los pares de nombre y valores recibidos por el método POST y GET

Prototipo:

int getFormNameValue(chstrarray & names, chstrarray & values);

Parámetros:

Names:arreglo de referencia que contendrá los nombres

Values:arreglo de referencia que contendrá los valores

Valor de retorno:

Entero que representa el número de pares recibidos

Class CServer Server: se utiliza para control de flujo de información del CGI

CServer::URLEncode:codifica caracteres utilizando las reglas URL

Prototipo:

chchar *URLEncode(chchar *in);

Parámetros:

In: string a codificar

Valor de retorno:

String que contiene la cadena codificada utilizando URL

c) Funciones para graficar

Class CPlot Plot: crea objeto Plot de la clase CPlot

CPlot::title: define el título del gráfico

Prototipo:

void title(string_t title);

Parámetro:

title: string que contiene el título deseado

Valor de retorno:

Ninguno

CPlot::legend

Prototipo:

void legend(string_t legend, int num);

Parámetro:

legend: string que contiene la leyenda

num: conjunto datos al que se le agrega la leyenda

Valor de retorno:

Ninguno

CPlot::outputType: define el tipo de salida del gráfico

Prototipo:

void outputType(PLOT_OUTPUTTYPE_STREAM, terminal)

Parámetro:

terminal: el tipo de terminal soportada (en la aplicación se utiliza “png color”)

Valor de retorno:

Ninguno

CPlot::plotting: despliega el gráfico

Prototipo:

void plotting();

Parámetro:

Ninguno

Valor de retorno:

Ninguno

CPlot::size:cambia el tamaño del gráfico

Prototipo:

void size(double x_scale, double y_scale);

Parámetro:

x_scale: factor de escalamiento del eje x

y_scale:factor de escalamiento del eje y

Valor de retorno:

Ninguno

CPlot::legendLocation: determina la localización de la leyenda

Prototipo:

void legendLocation(doble x, double y, …/*[double z]*/);

Parámetro:

x: coordenada x de la localización de la leyenda

y: coordenada y de la localización de la leyenda

Valor de retorno:

Ninguno

Código Fuente

modulo1.ch

```
#!/bin/ch
#include <cgi.h>
int main() {
  int  i, *kp, *tau, *kp2, *tau2, *kp3, *wn, *chi;
  chstrarray name[4],values[4];
  char *modelo;
  class CResponse Response;
  class CRequest Request;
  class CServer Server;
  Response.setContentType("text/html");

  Response.begin();
  /*titulo de la respuesta*/
  Response.title("Respuesta de la planta");
  /*limpia el buffer*/
  setbuf(stdout,NULL);
  /*Imprime el titulo de la respuesta*/
  printf("<H1>Respuesta de la planta</H1>\n");
  modelo=Request.getForm("modelo");

  /********switch para crear el modelo deseado**************/

  switch (modelo){
        case "primerorden":
        printf("Seleccion&oacute; el modelo de primer orden</p>");
        kp=Request.getForm("kp");
        tau=Request.getForm("tau");
        name[0]="kp";
        name[1]="tau1";
        name[2]="modelo";
        values[0]=kp;
        values[1]=tau;
        values[2]=modelo;
        printf("<table border=\"2\" align=\"center\" cellspacing=\"0\"
cellpadding=\"20\" bgcolor=\"#0a1e42\">");
        printf("<tr><td>");

        printf("<center><img src=\"modulo1_step.ch");

        /*paso de informaciÃ³n a modulo1_step.ch*/
        for(i=0;i<3;i++){
               putc(i==0 ? '?' : '&', stdout);
               fputs(Server.URLEncode(name[i]),stdout);
               putc('=',stdout);
               fputs(Server.URLEncode(values[i]),stdout);
               }
```

```
        printf("\">\n</center>\n");
        printf("</td>");
         printf("<td>");

        /*paso de informaciÃ³n a modulo1_pzmap.ch*/
         printf("<center><img src=\"modulo1_pzmap.ch");
         for(i=0;i<3;i++){
                putc(i==0 ? '?' : '&', stdout);
                fputs(Server.URLEncode(name[i]),stdout);
                putc('=',stdout);
                fputs(Server.URLEncode(values[i]),stdout);
         }
         printf("\">\n</center>\n");
        printf("</td>");
         printf("</tr>");
         printf("</table>");

         break;


        case "polodoble":
         printf("Seleccion&oacute; el modelo de polo doble</p>");
        kp2=Request.getForm("kp2");
         tau2=Request.getForm("tau2");
         name[0]="kp2";
         name[1]="tau2";
         name[2]="modelo";
         values[0]=kp2;
         values[1]=tau2;
         values[2]=modelo;
         printf("<table border=\"2\" align=\"center\" cellspacing=\"0\"
cellpadding=\"20\" bgcolor=\"#0a1e42\">");
         printf("<tr><td>");
        printf("<center><img src=\"/ch/modulo1_step.ch");

        /*paso de informaciÃ³n a modulo1_step.ch*/
        for(i=0;i<3;i++){
                putc(i==0 ? '?' : '&', stdout);
                fputs(Server.URLEncode(name[i]),stdout);
                putc('=',stdout);
                fputs(Server.URLEncode(values[i]),stdout);
                }
        printf("\">\n</center>\n");
        printf("</td>");
         printf("<td>");
         printf("<center><img src=\"/ch/modulo1_pzmap.ch");

        /*paso de informaciÃ³n a modulo1_pzmap.ch*/
         for(i=0;i<3;i++){
                putc(i==0 ? '?' : '&', stdout);
                fputs(Server.URLEncode(name[i]),stdout);
                putc('=',stdout);
```

```
                fputs(Server.URLEncode(values[i]),stdout);
        }
        printf("\">\n</center>\n");
        printf("</td>");
        printf("</tr>");
        printf("</table>");
        break;


        case "segundoorden":
        printf("Seleccion&oacute; el modelo de segundo orden</p>");
        kp3=Request.getForm("kp3");
        wn=Request.getForm("w");
        chi=Request.getForm("chi");
        name[0]="kp3";
        name[1]="w";
        name[2]="damp";
        name[3]="modelo";
        values[0]=kp3;
        values[1]=wn;
        values[2]=chi;
        values[3]=modelo;
        printf("<table border=\"2\" align=\"center\" cellspacing=\"0\"
cellpadding=\"20\" bgcolor=\"#0a1e42\">");
        printf("<tr><td>");
        printf("<center><img src=\"/ch/modulo1_step.ch");

        /*paso de informaciÃ³n a modulo1_step.ch*/
        for(i=0;i<4;i++){
                putc(i==0 ? '?' : '&', stdout);
                fputs(Server.URLEncode(name[i]),stdout);
                putc('=',stdout);
                fputs(Server.URLEncode(values[i]),stdout);
                }
                printf("\">\n</center>\n");

        printf("</td>");
        printf("<td>");

        /*paso de informaciÃ³n a modulo1_pzmap.ch*/
        printf("<center><img src=\"/ch/modulo1_pzmap.ch");
        for(i=0;i<4;i++){       /* pass system information to plot.ch */
                putc(i==0 ? '?' : '&', stdout);
                fputs(Server.URLEncode(name[i]),stdout);
                putc('=',stdout);
                fputs(Server.URLEncode(values[i]),stdout);
        }
        printf("\">\n</center>\n");
        printf("</td>");
        printf("</tr>");
        printf("</table>");
        break;
```

```
        }


    /*fin de la respuesta*/
    Response.end();
}
```

modulo1_step.ch

```
#!/bin/ch
#include <cgi.h>
#include <control.h>
#include <chplot.h>

int main() {
   /*declaración de variables*/
   int *kp,*tau,*damp;
   chstrarray name[4],value[4];
   char *modelo;
   class CResponse Response;
   class CRequest Request;
   /*plot será un objeto tipo CPlot*/
   class CPlot plot;
   /*sys será un objeto CControl*/
   class CControl sys;
   /*vectores para de declaración de la función de transferencia*/
   double den[3]={0,0,0};
   double num[1]={0};

   /*obtención del parámetro modelo*/
   modelo=Request.getForm("modelo");

   /*asignación de valores a num, y den según el modelo*/
   switch (modelo){
        case "primerorden":
        kp=Request.getForm("kp");
        tau=Request.getForm("tau1");
        num[0]=atof(kp);
        den[1]=atof(tau);
        den[2]=1;
        break;

        case "polodoble":
        kp=Request.getForm("kp2");
        tau=Request.getForm("tau2");
        num[0]=atof(kp);
        den[0]=atof(tau)*atof(tau);
        den[1]=2*atof(tau);
        den[2]=1;
        break;

        case "segundoorden":
        kp=Request.getForm("kp3");
        /*la frecuencia natural la llamaremos tau*/
        tau=Request.getForm("w");
        damp=Request.getForm("damp");
        num[0]=atof(kp)*atof(tau)*atof(tau);
        den[0]=1;
```

```
        den[1]=2*atof(damp)*atof(tau);
        den[2]=atof(tau)*atof(tau);
        break;
        }

    /*creación de la función de transferencia*/
    sys.model("tf", num, den);


    /*creación de respuesta tipo imagen*/
    Response.setContentType ("image/png");
    Response.begin();

    /*habilita la grilla*/
    sys.grid(1);
    sys.step(&plot,NULL,NULL,NULL);

    /*gráfica de la respuesta al escalón*/
    plot.title("Respuesta al escalon");

    /*títulos del gráfico*/
    /*forma de enviar la imagen para que pueda ser interpretada por el
browser*/
    plot.label(PLOT_AXIS_X, "Tiempo (s)");
    plot.label(PLOT_AXIS_Y, "Amplitud");
    /*grafico tipo png*/
    plot.outputType(PLOT_OUTPUTTYPE_STREAM, "png color");
    plot.size(0.7,0.7);

    /*graficación del gráfico con las características añadidas*/
    plot.plotting();

    /*fin de la respuesta*/
    Response.end();
}
```

modulo1_pzmap.ch

```
#!/bin/ch
#include <cgi.h>
#include <control.h>
#include <chplot.h>
#include <numeric.h>

int main() {

   /*declaración de variables*/

   int *kp,*tau,*damp,np,nz,i;
   chstrarray name[4],value[4];
   char *modelo;
   /*declaración de clases*/
   class CResponse Response;
   class CRequest Request;
   /*declaracion de zmap como un objeto CPlot*/
   class CPlot zmap;
   /*sys sera un objeto CControl, representará la función de
transferencia*/
   class CControl sys;
   array double den[3]={0,0,0};
   array double num[1]={0};

   /*obtiene el valor modelo*/
   modelo=Request.getForm("modelo");

   /*asignacion de valores a den y num según el caso*/
   switch (modelo){
        case "primerorden":
        kp=Request.getForm("kp");
        tau=Request.getForm("tau1");
        num[0]=atof(kp);
        den[1]=atof(tau);
        den[2]=1;
        break;

        case "polodoble":
        kp=Request.getForm("kp2");
        tau=Request.getForm("tau2");
        num[0]=atof(kp);
        den[0]=atof(tau)*atof(tau);
        den[1]=2*atof(tau);
        den[2]=1;
        break;

        case "segundoorden":
        kp=Request.getForm("kp3");
        /*la frecuencia natural la llamaremos tau*/
```

```
        tau=Request.getForm("w");
        damp=Request.getForm("damp");
        num[0]=atof(kp)*atof(tau)*atof(tau);
        den[0]=1;
        den[1]=2*atof(damp)*atof(tau);
        den[2]=atof(tau)*atof(tau);
       break;
        }

    /*creación de la función de transferencia*/
    sys.model("tf", num, den);

    /*obtención del número de polos y ceros*/
    np = sys.size('p');
    nz = sys.size('z');

  /*vectores de ceros y polos complejos*/
  array float complex z[nz];
  array float complex p[np];
  array float preal[np], pimag[np], zreal[nz], zimag[nz],
imagine[4]={0,0,0,0}, real[4]={0,0,0,0};

   /*asignación del vector de ceros a z y del vector de polos a p*/
  sys.tzero(z);
  sys.pole(p);


  /*división en parte real e imaginaria para vectores de polos y ceros*/
  for(i=0;i<np;i++){
  preal[i]=creal(p[i]);
  pimag[i]=cimag(p[i]);
  }

  for(i=0;i<nz;i++){
  zreal[i]=creal(z[i]);
  zimag[i]=cimag(z[i]);
  }

  /*establecimiento de los puntos límites del gráfico*/
  /*para esto se ubican los máximos y mínimos*/

  imagine[0]=pimag[maxloc(pimag)];
  imagine[1]=zimag[maxloc(zimag)];
  imagine[2]=pimag[minloc(pimag)];
  imagine[3]=zimag[minloc(zimag)];
  real[0]=preal[maxloc(preal)];
  real[1]=zreal[maxloc(zreal)];
  real[2]=preal[minloc(preal)];
  real[3]=zreal[minloc(zreal)];

  /*se crea una respuesta tipo imagen png*/
```

```
    Response.setContentType ("image/png");
    Response.begin();
    /*título de la gráfica*/
    zmap.title("Mapa de polos y ceros");

    /*establecimiento de los límites de los ejes*/
    zmap.axisRange(PLOT_AXIS_X, real[minloc(real)]-
0.5,real[maxloc(real)]+0.5);


    zmap.axisRange(PLOT_AXIS_Y, imagine[minloc(imagine)]-0.5,
imagine[maxloc(imagine)]+0.5);


    /*graficación de los polos como x*/
     for(i=0;i<np;i++){

       zmap.point(preal[i], pimag[i], 0);
       zmap.plotType(PLOT_PLOTTYPE_POINTS, i, 2 ,3);
     }

    /*graficación de los ceros como 0*/
     for(i=0;i<nz;i++){
     printf("real %f, imaginaria %f\n",zreal[i], zimag[i]);
     zmap.point(zreal[i], zimag[i], 0);
     zmap.plotType(PLOT_PLOTTYPE_POINTS, np+i,6,3);
     }
    /*titulos de los ejes*/
    zmap.label(PLOT_AXIS_X, "Eje Real");
    zmap.label(PLOT_AXIS_Y, "Eje Imaginario");

    /*grafico formato png*/
    /*forma de enviar la imagen para que pueda ser interpretada por el
browser*/
    zmap.outputType(PLOT_OUTPUTTYPE_STREAM, "png color");
    zmap.size(0.7,0.7);

    /*crea el gráfico con las condiciones establecidas*/
    zmap.grid(PLOT_ON);
    zmap.plotting();

    /*fin de la respuesta*/
    Response.end();
}
```

modulo2_1.ch

```
#!/bin/ch
#include <cgi.h>
#include <control.h>
#include <chplot.h>
int main() {

/**********declaración de variables***************/
  chstrarray name[13],values[13];
  char *modelo, *id;
/*********declaración de clases para CGI*****/
  class CResponse Response;
  class CRequest Request;
  class CServer Server;
/*********declaración de variables**********/
   int k, nz, np, i, loc,envio;



   class CPlot plot, plot2, plot3;
   class CControl sys, sys2;

   double den[5];
   double num[5], y[2001], t[2001], t2[2001], y3[2001], t3[2001];
   double kp, y25, t25, y50, t50, y75, t75, m, b, kpmodel, taumodel,
tmmodel, a, paso, s, s2, tfinal, bandera, tmoriginal;


  /*obtención del parámetro modelo*/
   modelo=Request.getForm("modelo");
 /*obtención del parámetro id (método de identificación)*/
   id=Request.getForm("id");

  switch (modelo){

 /*asignación a los vectores name y values en el caso de que el sistema*/
 /*se haya ingresado en su forma de división de polinomios de s*/
        case "division":

        name[0]="num4";
        name[1]="num3";
        name[2]="num2";
        name[3]="num1";
        name[4]="num0";
        name[5]="den4";
        name[6]="den3";
        name[7]="den2";
        name[8]="den1";
        name[9]="den0";
        name[10]="modelo";
```

```
name[11]="id";
values[0]=Request.getForm("num4");
values[1]=Request.getForm("num3");
values[2]=Request.getForm("num2");
values[3]=Request.getForm("num1");
values[4]=Request.getForm("num0");
values[5]=Request.getForm("den4");
values[6]=Request.getForm("den3");
values[7]=Request.getForm("den2");
values[8]=Request.getForm("den1");
values[9]=Request.getForm("den0");
values[10]=modelo;
values[11]=id;
envio=12;
break;

/*asignación a los vectores name y values en el caso de que el sistema*/
/*se haya ingresado en su forma de polos y ceros*/

case "pyz":
name[0]="z1";
name[1]="z2";
name[2]="z3";
name[3]="z4";
name[4]="p1";
name[5]="p2";
name[6]="p3";
name[7]="p4";
name[8]="kp";
name[9]="modelo";
name[10]="id";
name[11]="nz";
name[12]="np";
values[0]=Request.getForm("z1");
values[1]=Request.getForm("z2");
values[2]=Request.getForm("z3");
values[3]=Request.getForm("z4");
values[4]=Request.getForm("p1");
values[5]=Request.getForm("p2");
values[6]=Request.getForm("p3");
values[7]=Request.getForm("p4");
values[8]=Request.getForm("kp");
values[9]=modelo;
values[10]=id;
values[11]=Request.getForm("nz");
values[12]=Request.getForm("np");
envio=13;
break;

}
```

```
switch (modelo){

        /*construcción del modelo para el caso de división de polinomios
de s*/
        case "division":

        for(k=0; k<5; k++) {
                num[k]=atof(values[k]);
                den[k]=atof(values[k+5]);
                }
         sys.model("tf", num, den);
        break;
         /*construcción del modelo para el caso de polos y ceros*/
         case "pyz":
        nz=atoi(Request.getForm("nz"));
        /*construcción del vector de ceros*/
        array double complex zeros[nz];
        for(k=0; k<nz; k++) {
                zeros[k]=atoc(values[k]);
                 }

        np=atoi(Request.getForm("np"));
        /*construcción del vector de polos*/
         array double complex polos[np];
        for(k=0; k<np; k++) {
                polos[k]=atoc(values[k+4]);
                 }

         kp=atof(values[8]);
         sys.model("zpk", zeros, polos, kp);
        break;

         }

    /*RUTINA PARA CALCULO DE TIEMPOS REQUERIDOS*/

    /*respuesta al escalón asignada a y y tiempo asignado a t*/
    sys.step(&plot,y,t,NULL);
    tfinal=t[2000];
    sys.step(&plot,y,t,NULL,tfinal*5);
    /*obtención del par t25, y25 usando interpolación lineal*/
    y25=y[2000]*0.25;
    for(i=0;i < 2001; i++){
        loc=i;
        if(y[i] > y25){
                i=2001;
                }
        }
    m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
    b=y[loc]-m*t[loc];
    t25=(y25-b)/m;
```

```
 /*obtención del par t75, y75 usando interpolación lineal*/
 y75=y[2000]*0.75;
 for(i=0;i < 2001; i++){
     loc=i;
     if(y[i] > y75){
             i=2001;
             }
     }
 m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
 b=y[loc]-m*t[loc];
 t75=(y75-b)/m;

 /*obtención del par t50, y50 usando interpolación lineal*/
 y50=y[2000]*0.50;
 for(i=0;i < 2001; i++){
     loc=i;
     if(y[i] > y50){
             i=2001;
             }
     }
 m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
 b=y[loc]-m*t[loc];
 t50=(y50-b)/m;

 /*Fin de la rutina para calculo de tiempos*/


/*RUTINA PARA OBTENER MODELO*/

/*creación de los modelos identificados según sea el modelo deseado*/
/*en todos los casos se utiliza el método 123c de Alfaro*/

switch (id){
     case "primer":
     kpmodel=y[2000];
     taumodel=0.9102*(t75-t25);
     tmmodel=1.2620*t25-0.2620*t75;
     double num2[1]={kpmodel};
     double den2[2]={taumodel, 1};
     sys2.model("tf",num2,den2);
     sys2.step(&plot,NULL,t2,NULL);
     break;

     case "doble":
     kpmodel=y[2000];
     taumodel=0.5776*(t75-t25);
     tmmodel=1.5552*t25-0.5552*t75;
     double num3[1]={kpmodel};
     double den3[3]={taumodel*taumodel, 2*taumodel, 1};
     sys2.model("tf",num3,den3);
     sys2.step(&plot,NULL,t2,NULL);
```

```
        break;

        case "segundo":
        kpmodel=y[2000];
        a=(-0.6240*t25+0.9866*t50-0.3626*t75)/(0.3533*t25-
0.7036*t50+0.3503*t75);
        taumodel=(t75-t25)/(0.9866+0.7036*a);
        tmmodel=t75-(1.3421+1.3455*a)*taumodel;
        double num4[1]={kpmodel};
        double den4[3]={taumodel*a*taumodel, (taumodel+a*taumodel), 1};
        sys2.model("tf", num4, den4);
        sys2.step(&plot,NULL,t2,NULL);
        break;
    }

    if(tmmodel<0){
        tmoriginal=tmmodel;
        tmmodel=0;
        bandera=1;
        }

    sys2.step(&plot2,y3,t3,NULL,tfinal*1.25);

    /*rutina para graficar el tiempo muerto*/
    for(i=0;i<2001;i++){
        loc=i;
        if(t[i] >= tmmodel){
                i=2001;
                }
         }

    double t4[2001+loc];
    double y4[2001+loc];
    double yplanta2[2001+loc];
    paso=tmmodel/loc;

    for(i=0; i<loc; i++){
      t4[i]=paso*i;
      y4[i]=0;
      }
    for(i=0; i<2001; i++){
      t4[i+loc]=tmmodel+t3[i];
      y4[i+loc]=y3[i];
      }

    sys.grid(1);


        /*gráfico con la respuesta del modelo*/
        plotxy(t4, y4, "Respuesta al escalon", "Tiempo (s)", "Amplitud",
&plot3);
        sys.step(&plot3,yplanta2,NULL,NULL,t4[2001+loc-1]);
```

```
    /*calculo del indice de bondad del modelo*/
    s2=0;

    for(i=0;i<2001+loc;i++){
        s=y4[i]-yplanta2[i];
        s=s*s;
        s2=s2+s;
        }
```

```
  Response.setContentType("text/html");

  Response.begin();
  /*titulo de la respuesta*/
  Response.title("Identificaci&oacute;n de modelos");
  /*limpia el buffer*/
  setbuf(stdout,NULL);
  /*Imprime el titulo de la respuesta*/
  printf("<H1>Identificaci&oacute;n de modelos</H1>\n");

    switch (id) {
                /*selección de títulos de acuerdo al método de
identificación*/

                case "primer":
                 printf("<H2>Modelo de primer orden m&aacute;s tiempo
muerto</H2> </p>");
                 printf("<strong>Par&aacute;metros del
modelo</strong></p>");
                printf("kp=%f, tau=%f,
tm=%f</p>",kpmodel,taumodel,tmmodel);
                 break;

                 case "doble":
                 printf("<H2>Modelo de polo doble m&aacute;s tiempo
muerto</H2></p>");
                printf("<strong>Par&aacute;metros del
modelo</strong></p>");
                 printf("kp=%f, tau=%f,
tm=%f</p>",kpmodel,taumodel,tmmodel);
                 break;

                 case "segundo":
                printf("<H2>Modelo de segundo orden sobreamortiguado
m&aacute;s tiempo muerto</H2> </center>");
```

```
                printf("<strong>Par&aacute;metros del
modelo</strong></p>");
                printf("kp=%f, tau=%f, tm=%f
a=%f</p>",kpmodel,taumodel,tmmodel,a);
                break;
                 }

        if (bandera==1){
                printf("<span style=\"color: rgb(255, 0, 0);\">El tiempo
muerto calculado originalmente es %f</span></p>",tmoriginal);
                printf("<span style=\"color: rgb(255, 0, 0);\">Ha sido
ajustado a 0 por ser negativo</span>");
                }

        printf("<p><strong>Indice de bondad del modelo</strong></p>");
        printf("S2=%f",s2);

        printf("<center><img src=\"modulo2_image_1.ch");
        for(i=0;i<envio;i++){
                /* paso de la información a modulo2_image_1.ch*/
                /*de manera que se llame la función como si se usara el
método GET para cgis*/
                putc(i==0 ? '?' : '&', stdout);
                fputs(Server.URLEncode(name[i]),stdout);
                putc('=',stdout);
                fputs(Server.URLEncode(values[i]),stdout);
                }
        printf("\">\n</center>\n");
        /*fin de la respuesta*/
        Response.end();
}
```

modulo2_image_1.ch

```
#!/bin/ch
#include <cgi.h>
#include <control.h>
#include <chplot.h>

int main() {
   /*declaración de variables*/
   int k, nz, np, i, loc;
   chstrarray name,value;

   /*declaración de objetos de clase*/
   class CResponse Response;
   class CRequest Request;
   class CPlot plot, plot2, plot3;
   class CControl sys, sys2;

   /*declaración de variables*/
   double den[5];
   double num[5], y[2001], t[2001], t2[2001], y3[2001], t3[2001];
   double kp, y25, t25, y50, t50, y75, t75, m, b, kpmodel, taumodel,
tmmodel, a, paso, tfinal;
   char *modelo, *id;

   /*obtención de parámetros modelo e id*/
   modelo= Request.getForm("modelo");
   id=Request.getForm("id");

   /*asignación a name y value del nombre de cada valor y su valor
respectivo*/
   Request.getFormNameValue(name,value);

   switch (modelo){

        /*construcción del modelo para el caso de división de polinomios
de s*/
        case "division":

        for(k=0; k<5; k++) {
                num[k]=atof(value[k]);
                den[k]=atof(value[k+5]);
                }
        sys.model("tf", num, den);
        break;

        /*construcción del modelo para el caso de polos y ceros*/
        case "pyz":
        /*construcción del vector de ceros*/
        nz=atoi(Request.getForm("nz"));
        array double complex zeros[nz];
```

```
    for(k=0; k<nz; k++) {
            zeros[k]=atoc(value[k]);
             }

    /*construcción del vector de polos*/
    np=atoi(Request.getForm("np"));
    array double complex polos[np];
    for(k=0; k<np; k++) {
            polos[k]=atoc(value[k+4]);
             }

    kp=atof(value[8]);
    sys.model("zpk", zeros, polos, kp);
    break;

    }

/*respuesta al escalón asignada a "y" y tiempo asignado a "t"*/
sys.step(&plot,y,t,NULL);
tfinal=t[2000];
sys.step(&plot,y,t,NULL,tfinal*5);

/*obtención del par t25, y25 usando interpolación lineal*/
y25=y[2000]*0.25;
for(i=0;i < 2001; i++){
    loc=i;
    if(y[i] > y25){
            i=2001;
            }
    }
m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
b=y[loc]-m*t[loc];
t25=(y25-b)/m;


/*obtención del par t75, y75 usando interpolación lineal*/
y75=y[2000]*0.75;
for(i=0;i < 2001; i++){
    loc=i;
    if(y[i] > y75){
            i=2001;
            }
    }
m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
b=y[loc]-m*t[loc];
t75=(y75-b)/m;

/*obtención del par t50, y50 usando interpolación lineal*/
y50=y[2000]*0.50;
for(i=0;i < 2001; i++){
    loc=i;
    if(y[i] > y50){
```

```
                i=2001;
                }
        }
    m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
    b=y[loc]-m*t[loc];
    t50=(y50-b)/m;


    /*creación de una respuesta tipo imagen png*/
    Response.setContentType ("image/png");
    Response.begin();

    /*creación de los modelos identificados según sea el modelo deseado*/
    /*en todos los casos se utiliza el método 123c de Alfaro*/
    switch (id){
        case "primer":
        kpmodel=y[2000];
        taumodel=0.9102*(t75-t25);
        tmmodel=1.2620*t25-0.2620*t75;
        double num2[1]={kpmodel};
        double den2[2]={taumodel, 1};
        sys2.model("tf",num2,den2);
        sys2.step(&plot,NULL,t2,NULL);
        break;

        case "doble":
        kpmodel=y[2000];
        taumodel=0.5776*(t75-t25);
        tmmodel=1.5552*t25-0.5552*t75;
        double num3[1]={kpmodel};
        double den3[3]={taumodel*taumodel, 2*taumodel, 1};
        sys2.model("tf",num3,den3);
        sys2.step(&plot,NULL,t2,NULL);
        break;

        case "segundo":
        kpmodel=y[2000];
        a=(-0.6240*t25+0.9866*t50-0.3626*t75)/(0.3533*t25-
0.7036*t50+0.3503*t75);
        taumodel=(t75-t25)/(0.9866+0.7036*a);
        tmmodel=t75-(1.3421+1.3455*a)*taumodel;
        double num4[1]={kpmodel};
        double den4[3]={taumodel*a*taumodel, (taumodel+a*taumodel), 1};
        sys2.model("tf", num4, den4);
        sys2.step(&plot,NULL,t2,NULL);
        break;
    }

    if(tmmodel<0)tmmodel=0;
    sys2.step(&plot2,y3,t3,NULL,tfinal*1.25);
    /*rutina para graficar el tiempo muerto*/
    for(i=0;i<2001;i++){
```

```
        loc=i;
        if(t[i] >= tmmodel){
                i=2001;
                }
         }

    double t4[2001+loc];
    double y4[2001+loc];
    double yplanta2[2001+loc];
    paso=tmmodel/loc;

    for(i=0; i<loc; i++){
      t4[i]=paso*i;
      y4[i]=0;
      }
    for(i=0; i<2001; i++){
      t4[i+loc]=tmmodel+t3[i];
      y4[i+loc]=y3[i];
      }

    sys.grid(1);


        /*gráfico con la respuesta del modelo*/
        plotxy(t4, y4, "Respuesta al escalon", "Tiempo (s)", "Amplitud",
&plot3);
        sys.step(&plot3,yplanta2,NULL,NULL,t4[2001+loc-1]);




    plot3.legend("Modelo",0);
    plot3.legend("Planta",1);


    /*se establecen las propiedades de la gráfica de manera que pueda ser
interpretada por el browser*/
    plot3.outputType(PLOT_OUTPUTTYPE_STREAM, "png color");


    plot3.plotting();

    Response.end();
}
```

Modulo3_1.ch

```
#!/bin/ch
#include <cgi.h>
int main() {
  /*declaraciÃ³n de variables*/
  int  i, k;
  chstrarray name[13],values[13];
  char *modelo, *tool;
  class CResponse Response;
  class CRequest Request;
  class CServer Server;
  Response.setContentType("text/html");

  Response.begin();
  /*titulo de la respuesta*/
  Response.title("An&aacute;lisis de estabilidad");
  /*limpia el buffer*/
  setbuf(stdout,NULL);
  /*Imprime el titulo de la respuesta*/
  printf("<H1>An&aacute;lisis de estabilidad</H1>\n");

  /*obtenciÃ³n del parÃ¡metro modelo y la herramienta (tool)*/
  modelo=Request.getForm("modelo");
  tool=Request.getForm("tool");

  /*switch para crear la funciÃ³n de transferencia segÃºn modelo*/
  switch (modelo){

        case "division":

        name[0]="num4";
        name[1]="num3";
        name[2]="num2";
        name[3]="num1";
        name[4]="num0";
        name[5]="den4";
        name[6]="den3";
        name[7]="den2";
        name[8]="den1";
        name[9]="den0";
        name[10]="modelo";
        name[11]="tool";
        values[0]=Request.getForm("num4");
        values[1]=Request.getForm("num3");
        values[2]=Request.getForm("num2");
        values[3]=Request.getForm("num1");
        values[4]=Request.getForm("num0");
        values[5]=Request.getForm("den4");
        values[6]=Request.getForm("den3");
        values[7]=Request.getForm("den2");
```

```
        values[8]=Request.getForm("den1");
        values[9]=Request.getForm("den0");
         values[10]=modelo;
         values[11]=tool;
         k=12;
        break;

        case "pyz":
        name[0]="z1";
         name[1]="z2";
         name[2]="z3";
         name[3]="z4";
         name[4]="p1";
         name[5]="p2";
        name[6]="p3";
        name[7]="p4";
        name[8]="kp";
        name[9]="modelo";
         name[10]="tool";
         name[11]="nz";
         name[12]="np";
         values[0]=Request.getForm("z1");
         values[1]=Request.getForm("z2");
         values[2]=Request.getForm("z3");
        values[3]=Request.getForm("z4");
         values[4]=Request.getForm("p1");
         values[5]=Request.getForm("p2");
         values[6]=Request.getForm("p3");
        values[7]=Request.getForm("p4");
        values[8]=Request.getForm("kp");
        values[9]=modelo;
         values[10]=tool;
         values[11]=Request.getForm("nz");
         values[12]=Request.getForm("np");
        k=13;
         break;
         }

/*impresiÃ³n de tÃtulo segÃºn la herramienta seleccionada*/

         switch (tool) {

                case "lgr":
                 printf("<center> <H2>Gr&aacute;fica del lugar");
                printf(" geom&eacute;trico de");
                printf(" las ra&iacute;ces</H2> </center></p>");
                 break;

                 case "bode":
                 printf("<center> <H2>Gr&aacute;fico de Bode </H2>
</center></p>");
                 break;
```

```
                case "nyquist":
                printf("<center> <H2>Gr&aacute;fica de Nyquist</H2>
</center></p>");
                break;
                 }

/*paso de informaciÃ³n hacia modulo3_image_1.ch para creaciÃ³n de
imagen*/
        printf("<center><img src=\"modulo3_image_1.ch");
         for(i=0;i<k;i++){
                putc(i==0 ? '?' : '&', stdout);
                fputs(Server.URLEncode(name[i]),stdout);
                putc('=',stdout);
                fputs(Server.URLEncode(values[i]),stdout);
                }
         printf("\">\n</center>\n");
        Response.end();
}
```

modulo3_image_1.ch

```
#!/bin/ch
#include <cgi.h>
#include <control.h>
#include <chplot.h>

int main() {

/*declaraciÃ³n de variables*/
   int k, nz, np;
   chstrarray name,value;

   class CResponse Response;
   class CRequest Request;
   class CPlot plot;
   class CControl sys;

   double den[5];
   double num[5];
   double kp;
   char *modelo, *tool;

/*obtenciÃ³n de parÃ¡metros modelo y tool*/
   modelo= Request.getForm("modelo");
   tool=Request.getForm("tool");
/*asignaciÃ³n a name y value de los nombres y valores de los parÃ¡metros
recibidos*/
   Request.getFormNameValue(name,value);

/*switch para crear el modelo segÃºn el caso*/
   switch (modelo){

        case "division":

        for(k=0; k<5; k++) {
                num[k]=atof(value[k]);
                den[k]=atof(value[k+5]);
                }
         sys.model("tf", num, den);
        break;

         case "pyz":
        nz=atoi(Request.getForm("nz"));
        array double complex zeros[nz];
        for(k=0; k<nz; k++) {
                zeros[k]=atoc(value[k]);
                 }

        np=atoi(Request.getForm("np"));
         array double complex polos[np];
```

```
        for(k=0; k<np; k++) {
                polos[k]=atoc(value[k+4]);
                 }

        kp=atof(value[8]);
        sys.model("zpk", zeros, polos, kp);
        break;

        }

/*creaciÃ³n e inicio de respuesta tipo imagen*/
    Response.setContentType ("image/png");
    Response.begin();

/*creaciÃ³n de imagen segÃºn herramienta seleccionada*/

    switch (tool){
        case "bode":
        sys.margin(&plot,NULL,NULL,NULL,NULL);
        break;

        case "lgr":
        sys.rlocus(&plot,NULL,NULL);
        break;

        case "nyquist":
        sys.nyquist(&plot,NULL,NULL,NULL);
        break;
    }

    /*formato de imagen png para que sea interpretada por el browser*/
    plot.outputType(PLOT_OUTPUTTYPE_STREAM, "png color");
    plot.plotting();
    Response.end();
}
```

modulo4_1.ch

```
#!/bin/ch
#include <cgi.h>
int main() {
  int  i, k;
  chstrarray name[16],values[16];
  char *control, *modelo;
  class CResponse Response;
  class CRequest Request;
  class CServer Server;
  Response.setContentType("text/html");

  Response.begin();
  /*titulo de la respuesta*/
  Response.title("Modos de control");
  /*limpia el buffer*/
  setbuf(stdout,NULL);
  /*Imprime el titulo de la respuesta*/
  printf("<H1>Respuesta del sistema de control</H1>\n");

  /*obtenciÃ³n de los parÃ¡metos modelo y control (tipo de controlador)*/
  modelo=Request.getForm("modelo");
  control=Request.getForm("control");

  /*creaciÃ³n del sistema segÃºn el modelo*/


        switch(modelo){

        case "division":

        name[0]="num4";
        name[1]="num3";
        name[2]="num2";
        name[3]="num1";
        name[4]="num0";
        name[5]="den4";
        name[6]="den3";
        name[7]="den2";
        name[8]="den1";
        name[9]="den0";
        name[10]="control";
        name[11]="modelo";

        values[0]=Request.getForm("num4");
        values[1]=Request.getForm("num3");
        values[2]=Request.getForm("num2");
        values[3]=Request.getForm("num1");
        values[4]=Request.getForm("num0");
        values[5]=Request.getForm("den4");
```

```
        values[6]=Request.getForm("den3");
        values[7]=Request.getForm("den2");
        values[8]=Request.getForm("den1");
        values[9]=Request.getForm("den0");
        values[10]=control;
        values[11]=modelo;
        k=12;
        break;

        case "pyz":
        name[0]="z1";
        name[1]="z2";
        name[2]="z3";
        name[3]="z4";
        name[4]="p1";
        name[5]="p2";
        name[6]="p3";
        name[7]="p4";
        name[8]="kp";
        name[9]="modelo";
        name[10]="control";
        name[11]="nz";
        name[12]="np";
        values[0]=Request.getForm("z1");
        values[1]=Request.getForm("z2");
        values[2]=Request.getForm("z3");
        values[3]=Request.getForm("z4");
        values[4]=Request.getForm("p1");
        values[5]=Request.getForm("p2");
        values[6]=Request.getForm("p3");
        values[7]=Request.getForm("p4");
        values[8]=Request.getForm("kp");
        values[9]=modelo;
        values[10]=control;
        values[11]=Request.getForm("nz");
        values[12]=Request.getForm("np");
        k=13;
        break;
        }

        /*asignaciÃ³n del parÃ¡metro control segÃºn el controlador
seleccionado*/

        switch (control) {

               case "p":
                printf("<center> <H2>Respuesta utilizando un controlador
P</H2> </center></p>");
                name[k]="kcp";
                values[k]=Request.getForm("kcp");
                k=k+1;
                break;
```

```
                case "pi":
                printf("<center> <H2>Respuesta utilizando un controlador
PI</H2> </center></p>");
                name[k]="kcpi";
                name[k+1]="tipi";
                values[k]=Request.getForm("kcpi");
                values[k+1]=Request.getForm("tipi");
                k=k+2;
               break;

                case "pd":
               printf("<center> <H2>Respuesta utilizando un controlador
PD</H2> </center></p>");
               name[k]="kcpd";
               name[k+1]="tdpd";
               values[k]=Request.getForm("kcpd");
                values[k+1]=Request.getForm("tdpd");
               k=k+2;
               break;

               case "pid":
                name[k]="kcpid";
                name[k+1]="tipid";
                name[k+2]="tdpid";
                values[k]=Request.getForm("kcpid");
                values[k+1]=Request.getForm("tipid");
                values[k+2]=Request.getForm("tdpid");
                printf("<center> <H2>Respuesta utilizando un controlador
PID</H2> </center></p>");
               k=k+3;
               break;
               }


       printf("<table border=\"2\" align=\"center\" cellspacing=\"0\"
cellpadding=\"20\" bgcolor=\"#0a1e42\">");
        printf("<tr><td>");
       printf("<center><img src=\"modulo4_servo_1.ch");
       /*envÃo de informaciÃ³n a modulo4_servo_1.ch */

       for(i=0;i<k;i++){
               putc(i==0 ? '?' : '&', stdout);
               fputs(Server.URLEncode(name[i]),stdout);
               putc('=',stdout);
               fputs(Server.URLEncode(values[i]),stdout);
               }
               printf("\">\n</center>\n");

       printf("</td>");
       printf("<td>");
       /*envÃo de informaciÃ³n a modulo4_regulador_1.ch */
```

```
        printf("<center><img src=\"modulo4_regulador_1.ch");
        for(i=0;i<k;i++){
                putc(i==0 ? '?' : '&', stdout);
                fputs(Server.URLEncode(name[i]),stdout);
                putc('=',stdout);
                fputs(Server.URLEncode(values[i]),stdout);
        }
        printf("\">\n</center>\n");
        printf("</td>");
        printf("</tr>");
        printf("</table>");

        Response.end();
}
```

modulo4_servo_1.ch

```
#!/bin/ch
#include <cgi.h>
#include <control.h>
#include <chplot.h>

int main() {

   /*declaraciÃ³n de clases y variables*/
   int k, np, nz;
   chstrarray name,value;
   class CResponse Response;
   class CRequest Request;
   class CPlot plot;
   char *control, *kc1, *ti1, *td1, *modelo;
   double kc, ti, td, kp;
   double array den[5] = {0,0,0,0,0}, num[5] = {0,0,0,0,0}, numcontrol[3]
= {0,0,0}, dencontrol[3] = {0,0,0}, numretro[1]={1}, denretro[1]={1};
   class CControl sys, controlador, retro, *sys2, *sys3;

   /*asignaciÃ³n a name y value de los valores recibidos*/
   Request.getFormNameValue(name,value);

   /*asignaciÃ³n a control de la variable control recibida*/
   control=Request.getForm("control");
   /*asignaciÃ³n a modelo de la variable modelo  recibida*/
   modelo=Request.getForm("modelo");

   /*switch para crear el modelo*/
   switch (modelo){

        case "division":

        for(k=0; k<5; k++) {
                num[k]=atof(value[k]);
                den[k]=atof(value[k+5]);
                }
         sys.model("tf", num, den);
        break;

         case "pyz":
        nz=atoi(Request.getForm("nz"));
        array double complex zeros[nz];
        for(k=0; k<nz; k++) {
                zeros[k]=atoc(value[k]);
                 }

        np=atoi(Request.getForm("np"));
         array double complex polos[np];
        for(k=0; k<np; k++) {
```

```
            polos[k]=atoc(value[k+4]);
             }

        kp=atof(value[8]);
        sys.model("zpk", zeros, polos, kp);
       break;

       }

    /*switch para crear el numerador y denominador de la funcion de
transferencia del controlador*/
    switch (control){

       case "p":
        kc1=Request.getForm("kcp");
        kc=atof(kc1);
        numcontrol[2] = kc;
        dencontrol[2] = 1;
        break;

        case "pi":
        kc1=Request.getForm("kcpi");
        ti1=Request.getForm("tipi");

        kc=atof(kc1);
        ti=atof(ti1);
        numcontrol[1]=kc*ti;
        numcontrol[2]=kc;
        dencontrol[1]=ti;
        break;

        case "pd":
        kc1=Request.getForm("kcpd");
        td1=Request.getForm("tdpd");

        kc=atof(kc1);
        td=atof(td1);

       numcontrol[1]=kc*td*(1+0.1);
       numcontrol[0]=kc;
       dencontrol[1]=0.1*td;
       dencontrol[0]=1;
        break;

       case "pid":
        kc1=Request.getForm("kcpid");
        ti1=Request.getForm("tipid");
       td1=Request.getForm("tdpid");

       kc=atof(kc1);
        ti=atof(ti1);
        td=atof(td1);
```

```
        numcontrol[2]=kc;
        numcontrol[1]=kc*(ti+0.1*td);
        numcontrol[0]=kc*ti*td*(1+0.1);


       dencontrol[2]=0;
        dencontrol[1]=ti;
        dencontrol[0]=ti*0.1*td;

        break;

        }
    /*creaciÃ³n de la funciÃ³n de transferencia del controlador*/
    controlador.model("tf", numcontrol, dencontrol);

    /* conecta el controlador y la planta en serie */
    sys2=controlador.series(&sys);

    /* la retroalimentacion tiene ganancia unitaria */
    retro.model("tf", numretro, denretro);

    /*creaciÃ³n del lazo de retroalimentaciÃ³n*/
    sys3=sys2->feedback(&retro);

    /*creaciÃ³n e inicio de respuesta tipo imagen*/
    Response.setContentType ("image/png");
    Response.begin();

    /*respuesta al escalÃ³n del servomecanismo*/
    sys3->step(&plot,NULL,NULL,NULL);



    /*propiedades del grÃ¡fico*/


    plot.title("Servomecanismo");

    /*formato de imagen tipo png*/
    plot.outputType(PLOT_OUTPUTTYPE_STREAM, "png color");
    plot.size(0.7,0.7);
    plot.grid(PLOT_ON);
    plot.plotting();

    Response.end();

}
```

modulo4_regulador_1.ch

```
#!/bin/ch
#include <cgi.h>
#include <control.h>
#include <chplot.h>
#include <numeric.h>

int main() {

   /*declaraciÃ³n de clases y variables*/
   int k, np, nz, ornum,orden;
   chstrarray name,value;
   class CResponse Response;
   class CRequest Request;
   class CPlot plot, plot2;
   char *control, *kc1, *ti1, *td1, *modelo;
   double kc, ti, td, kp;
   double array den[5] = {0,0,0,0,0}, num[5] = {0,0,0,0,0}, numcontrol[3]
= {0,0,0}, dencontrol[3] = {0,0,0},y[100], t3[100];
   class CControl sys, controlador, *sys3;

   /*asignaciÃ³n a name y value de los nombres y valores de los
parÃ¡metros recibidos*/
   Request.getFormNameValue(name,value);

   /*obtenciÃ³n de los parÃ¡metros control y modelo*/
   control=Request.getForm("control");
   modelo=Request.getForm("modelo");


   /*creaciÃ³n del sistema segÃºn modelo seleccionado*/

   switch (modelo){

        case "division":

        for(k=0; k<5; k++) {
                num[k]=atof(value[k]);
                den[k]=atof(value[k+5]);
                }
        sys.model("tf", num, den);
        break;

        case "pyz":
        nz=atoi(Request.getForm("nz"));
        array double complex zeros[nz];
        for(k=0; k<nz; k++) {
                zeros[k]=atoc(value[k]);
                 }
```

```
    np=atoi(Request.getForm("np"));
    array double complex polos[np];
    for(k=0; k<np; k++) {
           polos[k]=atoc(value[k+4]);
            }

    kp=atof(value[8]);
    sys.model("zpk", zeros, polos, kp);
    ornum=sys.size('n');
    orden=sys.size('d');
    double array numpz[ornum],denpz[orden];
    sys.tfdata(numpz,denpz);
    for(k=0;k<ornum;k++){
           num[4-k]=numpz[ornum-1-k];
           }
    for(k=0;k<orden;k++){
            den[4-k]=denpz[orden-1-k];
    }
    break;

    }

/*creaciÃ³n y conexiÃ³n del controlador deseado*/

switch (control){

    case "p":
    kc1=Request.getForm("kcp");
    kc=atof(kc1);
    numcontrol[2] = kc;
    dencontrol[2] = 1;

    controlador.model("tf", numcontrol, dencontrol);
    sys3=sys.feedback(&controlador);

    break;

    case "pi":
    kc1=Request.getForm("kcpi");
    ti1=Request.getForm("tipi");

    kc=atof(kc1);
    ti=atof(ti1);
    numcontrol[1]=kc*ti;
    numcontrol[2]=kc;
    dencontrol[1]=ti;

    controlador.model("tf", numcontrol, dencontrol);
    sys3=sys.feedback(&controlador);

    break;
```

```
    case "pd":
    kc1=Request.getForm("kcpd");
    td1=Request.getForm("tdpd");

    kc=atof(kc1);
    td=atof(td1);

    numcontrol[1]=kc*td*(1+0.1);
    numcontrol[0]=kc;
    dencontrol[1]=0.1*td;
    dencontrol[0]=1;


    controlador.model("tf",numcontrol,dencontrol);
    sys3=sys.feedback(&controlador);
    break;

    case "pid":
    kc1=Request.getForm("kcpid");
    ti1=Request.getForm("tipid");
    td1=Request.getForm("tdpid");

    kc=atof(kc1);
    ti=atof(ti1);
    td=atof(td1);

    numcontrol[2]=kc;
    numcontrol[1]=kc*(ti+0.1*td);
    numcontrol[0]=kc*ti*td*(1+0.1);


    dencontrol[2]=0;
    dencontrol[1]=ti;
    dencontrol[0]=ti*0.1*td;

    controlador.model("tf",numcontrol,dencontrol);
    sys3=sys.feedback(&controlador);
    break;

    }


/*inicio de respuesta tipo imagen*/
Response.setContentType ("image/png");
Response.begin();


/*graficaciÃ³n de la respuesta al escalÃ³n del regulador*/
sys3->step(&plot,NULL,NULL,NULL);

plot.title("Regulador");
```

```
    /*grafico en formato png para que sea interpretado por el browser*/
    plot.outputType(PLOT_OUTPUTTYPE_STREAM, "png");
    /*propiedades del grÃ¡fico*/
    plot.size(0.7,0.7);
    plot.grid(PLOT_ON);
    plot.plotting();

    Response.end();

}
```

modulo5.ch

```
#!/bin/ch
#include <cgi.h>
#include <control.h>
#include  <chplot.h>
int main() {

  /*declaraciÃ³n de variables y clases*/
  int  i, k,np,nz,loc,envio;
  chstrarray name[16],values[16], criterios[3];
  criterios[0]="ISE";
  criterios[1]="IAE";
  criterios[2]="ITAE";

  char *control, *modelo, *metodo;
  double num[5],den[5],y[2001],t[2001],pend[2000], kc[3]={0,0,0},
ti[3]={0,0,0},td[3]={0,0,0}, y25,t25,y50,t50,y75,t75,y63, t63,
m,p,tau,tm,kp,dt,tpm,ind, tfinal, tmoriginal, bandera=0;
  double a[3]={0,0,0}, b[3]={0,0,0}, c[3]={0,0,0}, d[3]={0,0,0},
e[3]={0,0,0}, f[3]={0,0,0}, g[3]={0,0,0}, h[3]={0,0,0}, j[3]={0,0,0};
  class CResponse Response;
  class CRequest Request;
  class CServer Server;
  class CControl sys,sys2;
  class CPlot plot;

  /*creaciÃ³n e inicio de respuesta tipo html*/
  Response.setContentType("text/html");

  Response.begin();
  /*titulo de la respuesta*/
  Response.title("Sintonizaci&oacute;n de controladores");
  /*limpia el buffer*/
  setbuf(stdout,NULL);
  /*Imprime el titulo de la respuesta*/
  printf("<H1>Sintonizaci&oacute;n de controladores</H1>\n");
  printf("</p><strong>P&aacute;ramentros del modelo
obtenido</strong></p>");


  /*obtenciÃ³n del parÃ¡metro modelo, control y mÃ©todo de
sintonizaciÃ³n*/
  modelo=Request.getForm("modelo");
  control=Request.getForm("control");
  metodo=Request.getForm("metodo");


  /*obtenciÃ³n de parÃ¡metros del modelo*/
        switch(modelo){
```

```
case "division":

name[0]="num4";
 name[1]="num3";
 name[2]="num2";
 name[3]="num1";
 name[4]="num0";
 name[5]="den4";
name[6]="den3";
name[7]="den2";
name[8]="den1";
name[9]="den0";
 name[10]="control";
 name[11]="modelo";
 name[12]="metodo";
 values[0]=Request.getForm("num4");
 values[1]=Request.getForm("num3");
 values[2]=Request.getForm("num2");
values[3]=Request.getForm("num1");
 values[4]=Request.getForm("num0");
 values[5]=Request.getForm("den4");
 values[6]=Request.getForm("den3");
values[7]=Request.getForm("den2");
values[8]=Request.getForm("den1");
values[9]=Request.getForm("den0");
 values[10]=control;
 values[11]=modelo;
 values[12]=metodo;
 envio=13;
 break;

case "pyz":
name[0]="z1";
 name[1]="z2";
 name[2]="z3";
 name[3]="z4";
 name[4]="p1";
 name[5]="p2";
name[6]="p3";
name[7]="p4";
name[8]="kp";
name[9]="modelo";
 name[10]="control";
 name[11]="nz";
 name[12]="np";
name[13]="metodo";
 values[0]=Request.getForm("z1");
 values[1]=Request.getForm("z2");
 values[2]=Request.getForm("z3");
values[3]=Request.getForm("z4");
 values[4]=Request.getForm("p1");
 values[5]=Request.getForm("p2");
```

```
        values[6]=Request.getForm("p3");
        values[7]=Request.getForm("p4");
        values[8]=Request.getForm("kp");
        values[9]=modelo;
        values[10]=control;
        values[11]=Request.getForm("nz");
        values[12]=Request.getForm("np");
        values[13]=metodo;
        envio=14;
        break;
        }

        /*creaciÃ³n del modelo segÃºn el caso*/

        switch (modelo){

                case "division":

                for(k=0; k<5; k++) {
                        num[k]=atof(values[k]);
                        den[k]=atof(values[k+5]);
                        }
                sys.model("tf", num, den);
                break;

                case "pyz":

                nz=atoi(Request.getForm("nz"));
                array double complex zeros[nz];
                for(k=0; k<nz; k++) {
                        zeros[k]=atoc(values[k]);
                        }

                np=atoi(Request.getForm("np"));
                array double complex polos[np];
                for(k=0; k<np; k++) {
                        polos[k]=atoc(values[k+4]);
                        }

                kp=atof(values[8]);
                sys.model("zpk", zeros, polos, kp);
                break;
        }

    /*curva de reacciÃ³n*/

    sys.step(&plot,y,t,NULL);
    tfinal=t[2000];
    sys.step(&plot,y,t,NULL,tfinal*5);

    /*calculo de tiempos requeridos para modelos usando interpolaciÃ³n
lineal*/
```

```
    y63=y[2000]*0.63;
    for(i=0;i < 2001; i++){
        loc=i;
        if(y[i] > y63){
                i=2001;
                }
        }
    m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
    p=y[loc]-m*t[loc];
    t63=(y63-p)/m;

    y25=y[2000]*0.25;
    for(i=0;i < 2001; i++){
        loc=i;
        if(y[i] > y25){
                i=2001;
                }
        }
    m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
    p=y[loc]-m*t[loc];
    t25=(y25-p)/m;



    y75=y[2000]*0.75;
    for(i=0;i < 2001; i++){
        loc=i;
        if(y[i] > y75){
                i=2001;
                }
        }
    m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
    p=y[loc]-m*t[loc];
    t75=(y75-p)/m;

    y50=y[2000]*0.50;
    for(i=0;i < 2001; i++){
        loc=i;
        if(y[i] > y50){
                i=2001;
                }
        }
    m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
    p=y[loc]-m*t[loc];
    t50=(y50-p)/m;


    /*creaciÃ³n de los modelos usando ecuaciones correspondientes al
mÃ©todo 123c de Alfaro*/
    /*tambiÃ©n se usa el mÃ©todo de la tangente para sintonizaciÃ³n de
Zieglers y Nichols*/
```

```
switch (metodo){
    case "zyn":
    kp=y[2000];
    tau=t63;
    /*rutina para identificar segun metodo de la tangente*/
    dt=t[1]-t[0];
    pend[0]=0;
    for(i=1;i<1999;i++){
            pend[i]=(y[i+1]-y[i-1])/(2*dt);
            }
    ind=maxloc(pend);

    tpm=t[ind];
    tm=(-1*y[ind])/pend[ind]+tpm;
    if(tm<0){
            tmoriginal=tm;
            tm=0;
            }
     printf("</p>kp= %f, tau= %f, tm= %f</p>",kp,tau,tm);
    break;

    /*solera utiliza un modelo de polo doble*/
     case "solerareg":
     kp=y[2000];
    tau=0.5776*(t75-t25);
     tm=1.5552*t25-0.5552*t75;
    if(tm<0){
            tmoriginal=tm;
            tm=0;
            }
    printf("</p>kp= %f, tau= %f, tm= %f</p>",kp,tau,tm);
     break;

    case "soleraservo":
    kp=y[2000];
    tau=0.5776*(t75-t25);
     tm=1.5552*t25-0.5552*t75;
    if(tm<0){
            tmoriginal=tm;
            tm=0;
            }
    printf("</p>kp= %f, tau= %f, tm= %f</p>",kp,tau,tm);
     break;

    /*en todos los demÃ¡s casos se usa un modelo de primer orden*/
    default:
     kp=y[2000];
     tau=0.9102*(t75-t25);
     tm=1.2620*t25-0.2620*t75;
    if(tm<0){
            tmoriginal=tm;
```

```
                tm=0;
                }
        printf("</p>kp= %f, tau= %f, tm= %f</p>",kp,tau,tm);
        break;
        }

    /*anuncio de ajuste en caso de que el tiempo muerto calculado sea
negativo*/
        if (tmoriginal<0){
                printf("<span style=\"color: rgb(255, 0, 0);\">El tiempo
muerto calculado originalmente es %f</span></p>",tmoriginal);
                printf("<span style=\"color: rgb(255, 0, 0);\">Ha sido
ajustado a 0 por ser negativo</span>");
                bandera=1;
                }

    /*determinaciÃ³n de los parÃ¡metros del controlador segÃºn el mÃ©todo
y tipo de controlador*/


     switch (control) {

                case "p":


                switch (metodo) {

                        case "zyn":
                         kc[0]=tau/(kp*tm);
                         printf("<h2>M&eacute;todo de Z y N </h2></p>");
                         printf("<strong>Par&aacute;metro del controlador
P</strong></p>");
                        printf("Kc = %f", kc[0]);
                        if(tm>tau){
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm&gt;tau</span></p>");
                                bandera=1;
                                }
                        break;

                         case "lopez":

                         printf ("<h2>M&eacute;todo de Lopez</h2></p>");
                         a[0]=1.410;
                        a[1]=0.9023;
                         a[2]=0.4897;
                         b[0]=-0.9170;
                         b[1]=-0.9850;
                        b[2]=-1.0850;
                        printf("<strong>Par&aacute;metro del controlador
P</strong></p>");
                        for(i=0;i<3;i++) {
```

```
                                kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
                                  fputs(criterios[i],stdout);
                                  printf(" Kc = %f </p>", kc[i]);
                                 }
                        if((tm/tau)<0.1){
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm/tau &lt; 0.1</span></p>");
                                bandera=1;
                                }
                        break;

                        default:
                         printf("<p>El m&eacute;todo seleccionado no
sintoniza controladores tipo P\n");
                          break;

                        }
                 break;

                case "pi":

                switch (metodo) {

                         case "zyn":
                         kc[0]=0.9*tau/(kp*tm);
                        ti[0]=3.33*tm;
                        printf("<h2>M&eacute;todo de Z y N </h2></p>");
                         printf("<strong>Par&aacute;metros del controlador
PI</strong></p>");
                        printf("Kc = %f</p>", kc[0]);
                         printf("Ti= %f</p>", ti[0]);
                        if(tm>tau){
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm&gt;tau</span></p>");
                                bandera=1;
                                }
                        break;

                        case "lopez":
                        printf ("<h2>M&eacute;todo de
L&oacute;pez</h2></p>");
                         a[0]=1.3050;
                        a[1]=0.9840;
                         a[2]=0.8590;
                         b[0]=-0.9600;
                         b[1]=-0.9860;
                        b[2]=-0.9770;
                        c[0]=2.0325;
                        c[1]=1.6447;
                         c[2]=1.4837;
                         d[0]=-0.7390;
                         d[1]=-0.7070;
```

```
                        d[2]=-0.6800;
                        printf("<strong>Par&aacute;metros del controlador
PI</strong></p>");
                        for(i=0;i<3;i++) {
                                kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
                                  ti[i]=c[i]*tau*pow(tm/tau, d[i]);
                                  fputs(criterios[i],stdout);
                                  printf("</p>Kc= %f Ti = %f </p>", kc[i],
ti[i]);
                                }
                        if((tm/tau)<0.1){
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm/tau &lt; 0.1</span></p>");
                                bandera=1;
                                }
                        break;

                        case "rovira":
                        printf ("<h2>M&eacute;todo de Rovira</h2></p>");
                        a[1]=0.7580;
                         a[2]=0.5860;
                        b[1]=-0.8610;
                        b[2]=-0.9160;
                        c[1]=1.0200;
                        c[2]=1.0300;
                        d[1]=-0.3230;
                        d[2]=-0.1650;
                        printf("<strong>Par&aacute;metros del controlador
PI</strong></p>");
                        for(i=1;i<3;i++) {
                                kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
                                  ti[i]=tau/(c[i]+d[i]*(tm/tau));
                                  fputs(criterios[i],stdout);
                                  printf("</p> Kc= %f Ti = %f </p>", kc[i],
ti[i]);
                                }
                        if((tm/tau)<0.1){
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm/tau &lt; 0.1</span></p>");
                                        bandera=1;
                                }
                        break;


                        case "arrietareg":
                         printf("<h2>M&eacute;todo de Arrieta para
reguladores</h2>");
                        a[1]=0.45;
                        a[2]=0.2607;
                         b[1]=0.6494;
                        b[2]=0.6470;
                         c[1]=-1.1251;
```

```
                        c[2]=-1.1055;
                        d[1]=-0.2551;
                        d[2]=-1.5926;
                        e[1]=1.8205;
                        e[2]=2.9191;
                        f[1]=0.4749;
                        f[2]=0.1789;

                        printf("<strong>Par&aacute;metros del controlador
PI</strong></p>");
                        for(i=1;i<3;i++) {
                                kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
                                  ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
                                  fputs(criterios[i],stdout);
                                  printf("</p> Kc= %f Ti = %f </p>", kc[i],
ti[i]);
                                }
                        if((tm/tau)<0.1){
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm/tau &lt; 0.1</span></p>");
                                bandera=1;
                                }
                        break;


                        case "arrietaservo":
                        printf("<h2>M&eacute;todo de Arrieta para
servomecanismos</h2>");
                        a[1]=0.2438;
                        a[2]=0.1140;
                         b[1]=0.5305;
                        b[2]=0.5131;
                         c[1]=-1.0299;
                        c[2]=-1.0382;
                        d[1]=0.9377;
                        d[2]=0.9953;
                        e[1]=0.4337;
                        e[2]=0.2073;
                        f[1]=0.8714;
                        f[2]=1.5246;

                        printf("<strong>Par&aacute;metros del controlador
PI</strong></p>");
                        for(i=1;i<3;i++) {
                                kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
                                  ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
                                  fputs(criterios[i],stdout);
                                  printf("</p> Kc= %f Ti = %f </p>", kc[i],
ti[i]);
                                }
                        if((tm/tau)<0.1){
```

```
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm/tau &lt; 0.1</span></p>");
                                bandera=1;
                                }

                        break;

                        case "solerareg":
                        printf("<h2>M&eacute;todo de Solera para
reguladores</h2>");
                        a[1]=0.1234;
                        a[2]=0;
                         b[1]=1.0640;
                        b[2]=0.9239;
                         c[1]=-0.9487;
                        c[2]=-0.8616;
                        d[1]=2.48;
                        d[2]=2.0230;
                        e[1]=0.0607;
                        e[2]=0.1336;
                        f[1]=2.0550;
                        f[2]=1.3590;

                        printf("<strong>Par&aacute;metros del controlador
PI</strong></p>");
                        for(i=1;i<3;i++) {
                                kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
                                  ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
                                  fputs(criterios[i],stdout);
                                  printf("</p> Kc= %f Ti = %f </p>", kc[i],
ti[i]);
                                }
                        if((tm/tau)<0.5){
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm/tau &lt; 0.5</span></p>");
                                bandera=1;
                                }
                        break;

                        case "soleraservo":
                        printf("<h2>M&eacute;todo de Solera para
servomecanismos</h2>");
                        if(tm/tau<=0.65){
                            a[1]=0;
                            b[1]=0.8629;
                            c[1]=-0.5814;
                            d[1]=0;
                            e[1]=2.1360;
                            f[1]=-0.0861;
                        }
                        else{
                            a[1]=0;
```

```
                        b[1]=0.8629;
                        c[1]=-0.5817;
                        d[1]=2.2440;
                        e[1]=0.0555;
                        f[1]=3.1930;
                    }

                    if(tm/tau<=0.35){
                        a[2]=0;
                        b[2]=0.6477;
                        c[2]=-0.4927;
                        d[2]=0;
                        e[2]=1.7270;
                        f[2]=-0.0429;
                    }
                    else{
                        a[2]=0;
                        b[2]=0.6477;
                        c[2]=-0.4927;
                        d[2]=1.7910;
                        e[2]=0.1012;
                        f[2]=1.772;
                    }



                    printf("<strong>Par&aacute;metros del controlador
PI</strong></p>");
                    for(i=1;i<3;i++) {
                            kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
                              ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
                              fputs(criterios[i],stdout);
                              printf("</p> Kc= %f Ti = %f </p>", kc[i],
ti[i]);
                            }
                    if((tm/tau)<0.5){
                            printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm/tau &lt; 0.5</span></p>");
                            bandera=1;
                            }
                    break;


                    }
            break;

            case "pid":

            switch (metodo) {

                    case "zyn":
                    kc[0]=2*tau/(kp*tm);
```

```
                        ti[0]=2.5*tm;
                        td[0]=0.4*tm;
                        printf("<h2>M&eacute;todo de Z y N </h2></p>");
                         printf("<strong>Par&aacute;metros del controlador
PID</strong></p>");
                        printf("Kc = %f</p>", kc[0]);
                         printf("Ti= %f</p>", ti[0]);
                        printf("Td= %f</p>", td[0]);
                        if(tm>tau){
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm&gt;tau</span>");
                                bandera=1;
                                }
                        break;

                        case "lopez":
                        printf ("<h2>M&eacute;todo de
L&oacute;pez</h2></p>");
                         a[0]=1.4950;
                        a[1]=1.4350;
                         a[2]=1.3570;
                         b[0]=-0.9450;
                         b[1]=-0.9210;
                        b[2]=-0.9470;
                        c[0]=0.9083;
                        c[1]=1.1390;
                         c[2]=1.1876;
                         d[0]=0.7710;
                         d[1]=0.7490;
                        d[2]=0.7380;
                        e[0]=0.5600;
                        e[1]=0.4820;
                        e[2]=0.3810;
                        f[0]=1.0060;
                        f[1]=1.1370;
                        f[2]=0.9950;
                        printf("<strong>Par&aacute;metros del controlador
PID</strong></p>");
                        for(i=0;i<3;i++) {
                                kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
                                  ti[i]=c[i]*tau*pow(tm/tau, d[i]);
                                  td[i]=e[i]*tau*pow(tm/tau, f[i]);
                                  fputs(criterios[i],stdout);
                                  printf("</p>Kc= %f Ti = %f Td= %f</p>",
kc[i], ti[i], td[i]);
                                }
                        if((tm/tau)<0.1){
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm/tau &lt; 0.1</span></p>");
                                bandera=1;
                                }
```

```
                        break;

                        case "rovira":
                        printf ("<h2>M&eacute;todo de Rovira</h2></p>");
                        a[1]=1.0860;
                         a[2]=0.9650;
                        b[1]=-0.8690;
                        b[2]=-0.8500;
                        c[1]=0.7400;
                        c[2]=1.7960;
                        d[1]=-0.1300;
                        d[2]=-0.1465;
                        e[1]=0.3480;
                        e[2]=0.3080;
                        f[1]=0.9140;
                        f[2]=0.9290;
                        printf("<strong>Par&aacute;metros del controlador
PID</strong></p>");
                        for(i=1;i<3;i++) {
                                kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
                                  ti[i]=tau/(c[i]+d[i]*(tm/tau));
                                  td[i]=e[i]*tau*pow(tm/tau, b[i]);
                                  fputs(criterios[i],stdout);
                                  printf("</p>Kc= %f Ti = %f Td = %f</p>",
kc[i], ti[i], td[i]);
                                }
                        if((tm/tau)<0.1){
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm/tau &lt; 0.1</span></p>");
                                bandera=1;
                                }
                        break;

                        case "arrietareg":
                         printf("<h2>M&eacute;todo de Arrieta para
reguladores</h2>");
                        a[1]=0.45;
                        a[2]=0.2607;
                         b[1]=0.6494;
                        b[2]=0.6470;
                         c[1]=-1.1251;
                        c[2]=-1.1055;
                        d[1]=-0.2551;
                        d[2]=-1.5926;
                        e[1]=1.8205;
                        e[2]=2.9191;
                        f[1]=0.4749;
                        f[2]=0.1789;
                        g[1]=-0.0003;
                        g[2]=-0.0053;
                        h[1]=0.3838;
                        h[2]=0.3695;
```

```
                        j[1]=0.9479;
                        j[2]=0.9286;

                        printf("<strong>Par&aacute;metros del controlador
PID</strong></p>");
                        for(i=1;i<3;i++) {
                                kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
                                  ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
                                td[i]=tau*(g[i]+h[i]*pow(tm/tau,j[i]));
                                  fputs(criterios[i],stdout);
                                  printf("</p> Kc= %f Ti = %f Td = %f</p>",
kc[i], ti[i],td[i]);
                                }
                        if((tm/tau)<0.1){
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm/tau &lt; 0.1</span></p>");
                                bandera=1;
                                }
                        break;

                        case "arrietaservo":
                        printf("<h2>M&eacute;todo de Arrieta para
servomecanismos</h2>");
                        a[1]=0.2438;
                        a[2]=0.1140;
                         b[1]=0.5305;
                        b[2]=0.5131;
                         c[1]=-1.0299;
                        c[2]=-1.0382;
                        d[1]=0.9377;
                        d[2]=0.9953;
                        e[1]=0.4337;
                        e[2]=0.2073;
                        f[1]=0.8714;
                        f[2]=1.5246;
                        g[1]=-0.0146;
                        g[2]=-0.0169;
                        h[1]=0.3500;
                        h[2]=0.3126;
                        j[1]=0.8100;
                        j[2]=0.7417;

                        printf("<strong>Par&aacute;metros del controlador
PID</strong></p>");
                        for(i=1;i<3;i++) {
                                kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
                                  ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
                                td[i]=tau*(g[i]+h[i]*pow(tm/tau,j[i]));
                                  fputs(criterios[i],stdout);
                                  printf("</p> Kc= %f Ti = %f Td = %f</p>",
kc[i], ti[i],td[i]);
                                }
```

```
                        if((tm/tau)<0.1){
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm/tau &lt; 0.1</span></p>");
                                bandera=1;
                                }
                        break;

                        case "solerareg":
                        printf("<h2>M&eacute;todo de Solera para
reguladores</h2>");
                        a[1]=0.6199;
                        a[2]=0.5361;
                         b[1]=1.1080;
                        b[2]=1.1210;
                         c[1]=-1.4810;
                        c[2]=-1.4470;
                        d[1]=-1.1940;
                        d[2]=-2.9570;
                        e[1]=2.8340;
                        e[2]=4.7350;
                        f[1]=0.2686;
                        f[2]=0.1556;
                        g[1]=-0.2729;
                        g[2]=-0.2507;
                        h[1]=1.0640;
                        h[2]=0.9522;
                        j[1]=0.3521;
                        j[2]=0.3039;

                        printf("<strong>Par&aacute;metros del controlador
PID</strong></p>");
                        for(i=1;i<3;i++) {
                                kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
                                  ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
                                td[i]=tau*(g[i]+h[i]*pow(tm/tau,j[i]));
                                  fputs(criterios[i],stdout);
                                  printf("</p> Kc= %f Ti = %f Td = %f</p>",
kc[i], ti[i],td[i]);
                                }
                        if((tm/tau)<0.5){
                                printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm/tau &lt; 0.5</span></p>");
                                bandera=1;
                                }
                        break;

                        case "soleraservo":
                        printf("<h2>M&eacute;todo de Solera para
servomecanismos</h2>");
                        if(tm/tau<=0.45){
                            a[1]=0;
                            b[1]=1.2460;
```

```
                    c[1]=-0.8050;
                    d[1]=2.1420;
                    e[1]=-0.1356;
                    f[1]=-0.2790;
                }
                else{
                    a[1]=0;
                    b[1]=1.2460;
                    c[1]=-0.8050;
                    d[1]=1.9630;
                    e[1]=0.0805;
                    f[1]=2.2110;
                }

                if(tm/tau<=0.65){
                    a[2]=0.3004;
                    b[2]=0.8471;
                    c[2]=-0.9218;
                    d[2]=1.6980;
                    e[2]=0.3713;
                    f[2]=0.2957;
                }
                else{
                    a[2]=0.3004;
                    b[2]=0.8471;
                    c[2]=-0.9218;
                    d[2]=1.9390;
                    e[2]=0.1486;
                    f[2]=1.593;
                }

                g[1]=0.4523;
                g[2]=0.4629;
                h[1]=0.2477;
                h[2]=0.1414;
                j[1]=0.9587;
                j[2]=1.275;

                printf("<strong>Par&aacute;metros del controlador
PID</strong></p>");
                for(i=1;i<3;i++) {
                        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
                          ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
                        td[i]=tau*(g[i]+h[i]*pow(tm/tau,j[i]));
                          fputs(criterios[i],stdout);
                          printf("</p> Kc= %f Ti = %f Td = %f</p>",
kc[i], ti[i],td[i]);
                        }
                if((tm/tau)<0.5){
                        printf("<span style=\"color: rgb(255, 0,
0);\">El m&eacute;todo no aplica ya que tm/tau &lt; 0.5</span></p>");
                        bandera=1;
```

```
                        }
                break;



                        }
        break;


        }


/*graficas en caso de que el mÃ©todo si aplique*/
if(bandera==0){



printf("<table border=\"2\" align=\"center\" cellspacing=\"0\"
cellpadding=\"20\" bgcolor=\"#0a1e42\">");
 printf("<tr><td>");
printf("<center><img src=\"modulo5_image_1.ch");

for(i=0;i<envio;i++){
        putc(i==0 ? '?' : '&', stdout);
        fputs(Server.URLEncode(name[i]),stdout);
        putc('=',stdout);
        fputs(Server.URLEncode(values[i]),stdout);
        }
         printf("\">\n</center>\n");

printf("</td>");
 printf("<td>");
 printf("<center><img src=\"modulo5_image_2.ch");
 /*envÃo de parÃ¡metros a modulo5_image_1.ch*/
 for(i=0;i<envio;i++){
        putc(i==0 ? '?' : '&', stdout);
        fputs(Server.URLEncode(name[i]),stdout);
        putc('=',stdout);
        fputs(Server.URLEncode(values[i]),stdout);
        }
 printf("\">\n</center>\n");
 printf("</td>");
 printf("</tr>");
 printf("</table>");
 }
 Response.end();
}
```

modulo5_image_1.ch

```
#!/bin/ch
#include <cgi.h>
#include <control.h>
#include  <chplot.h>
int main() {

  /*declaraciÃ³n de clases y variables*/
  int  i, k,np,nz,loc;
  chstrarray name[16],values[16], criterios[3];
  criterios[0]="ISE";
  criterios[1]="IAE";
  criterios[2]="ITAE";

  char *control, *modelo, *metodo;
  double num[5],den[5],y[2001],t[2001],pend[2000], kc[3]={0,0,0},
ti[3]={0,0,0},td[3]={0,0,0}, y25,t25,y50,t50,y75,t75,y63, t63,
m,p,tau,tm,kp,dt,tpm,ind, tfinal;
  double a[3]={0,0,0}, b[3]={0,0,0}, c[3]={0,0,0}, d[3]={0,0,0},
e[3]={0,0,0}, f[3]={0,0,0}, g[3]={0,0,0}, h[3]={0,0,0}, j[3]={0,0,0};
  double numretro[1]={1}, denretro[1]={1};
  class CResponse Response;
  class CRequest Request;
  class CServer Server;
  class CControl sys,control1,*serie, *lazo, sys4,retro;
  class CPlot plot,plot2;

  /*creaciÃ³n de una respuesta tipo imagen*/

  Response.setContentType("image/png");

  Response.begin();


  setbuf(stdout,NULL);                        /*limpia el buffer*/

  /*obtenciÃ³n de los parÃ¡metos modelo, control y mÃ©todo de
sintonizaciÃ³n*/

  modelo=Request.getForm("modelo");
  control=Request.getForm("control");
  metodo=Request.getForm("metodo");

  /*asignaciÃ³n a name y values de los valores recibidos*/

        switch(modelo){

        case "division":

        name[0]="num4";
```

```
name[1]="num3";
name[2]="num2";
name[3]="num1";
name[4]="num0";
name[5]="den4";
name[6]="den3";
name[7]="den2";
name[8]="den1";
name[9]="den0";
name[10]="control";
name[11]="modelo";
name[12]="metodo";
values[0]=Request.getForm("num4");
values[1]=Request.getForm("num3");
values[2]=Request.getForm("num2");
values[3]=Request.getForm("num1");
values[4]=Request.getForm("num0");
values[5]=Request.getForm("den4");
values[6]=Request.getForm("den3");
values[7]=Request.getForm("den2");
values[8]=Request.getForm("den1");
values[9]=Request.getForm("den0");
values[10]=control;
values[11]=modelo;
values[12]=metodo;
break;

case "pyz":
name[0]="z1";
name[1]="z2";
name[2]="z3";
name[3]="z4";
name[4]="p1";
name[5]="p2";
name[6]="p3";
name[7]="p4";
name[8]="kp";
name[9]="modelo";
name[10]="control";
name[11]="nz";
name[12]="np";
name[13]="metodo";
values[0]=Request.getForm("z1");
values[1]=Request.getForm("z2");
values[2]=Request.getForm("z3");
values[3]=Request.getForm("z4");
values[4]=Request.getForm("p1");
values[5]=Request.getForm("p2");
values[6]=Request.getForm("p3");
values[7]=Request.getForm("p4");
values[8]=Request.getForm("kp");
values[9]=modelo;
```

```
    values[10]=control;
    values[11]=Request.getForm("nz");
    values[12]=Request.getForm("np");
    values[13]=metodo;
    break;
    }


    /*creaciÃ³n de la planta*/

    switch (modelo){

            case "division":

            for(k=0; k<5; k++) {
                    num[k]=atof(values[k]);
                    den[k]=atof(values[k+5]);
                    }
            sys.model("tf", num, den);
            break;

            case "pyz":

            nz=atoi(Request.getForm("nz"));
            array double complex zeros[nz];
            for(k=0; k<nz; k++) {
                    zeros[k]=atoc(values[k]);
                    }

            np=atoi(Request.getForm("np"));
            array double complex polos[np];
            for(k=0; k<np; k++) {
                    polos[k]=atoc(values[k+4]);
                    }

            kp=atof(values[8]);
            sys.model("zpk", zeros, polos, kp);
            break;
    }

/*curva de reacciÃ³n*/
sys.step(&plot,y,t,NULL);
tfinal=t[2000];
sys.step(&plot,y,t,NULL,tfinal*5);

/*calculo de tiempos requeridos para obtener los modelos*/
y63=y[2000]*0.63;
for(i=0;i < 2001; i++){
    loc=i;
    if(y[i] > y63){
            i=2001;
            }
```

```
    }
m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
p=y[loc]-m*t[loc];
t63=(y63-p)/m;

y25=y[2000]*0.25;
for(i=0;i < 2001; i++){
    loc=i;
    if(y[i] > y25){
            i=2001;
            }
    }
m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
p=y[loc]-m*t[loc];
t25=(y25-p)/m;



y75=y[2000]*0.75;
for(i=0;i < 2001; i++){
    loc=i;
    if(y[i] > y75){
            i=2001;
            }
    }
m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
p=y[loc]-m*t[loc];
t75=(y75-p)/m;

y50=y[2000]*0.50;
for(i=0;i < 2001; i++){
    loc=i;
    if(y[i] > y50){
            i=2001;
            }
    }
m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
p=y[loc]-m*t[loc];
t50=(y50-p)/m;

/*identificaciÃ³n de modelos segÃºn el caso*/
/*tangente para Zieglers y Nichols*/
/*polo doble usando 123c de Alfaro para Solera*/
/*primer orden usando 123c de Alfaro en los demÃ¡s casos*/

switch (metodo){
    case "zyn":
    kp=y[2000];
    tau=t63;
    /*rutina para identificar segun metodo de la tangente*/
    dt=t[1]-t[0];
    pend[0]=0;
```

```
        for(i=1;i<1999;i++){
                pend[i]=(y[i+1]-y[i-1])/(2*dt);
                }
        ind=maxloc(pend);

        tpm=t[ind];
        tm=(-1*y[ind])/pend[ind]+tpm;

        break;

        case "solerareg":
        kp=y[2000];
        tau=0.5776*(t75-t25);
        tm=1.5552*t25-0.5552*t75;
        if(tm<0)tm=0;

        break;

        case "soleraservo":
        kp=y[2000];
        tau=0.5776*(t75-t25);
        tm=1.5552*t25-0.5552*t75;
        if(tm<0)tm=0;
        break;

        default:
        kp=y[2000];
        tau=0.9102*(t75-t25);
        tm=1.2620*t25-0.2620*t75;
        if(tm<0)tm=0;
        break;
        }

     /*creaciÃ³n de la funciÃ³n de transferencia del controlador y
     conexiÃ³n como servomecanismo*/
     /*se crean tambiÃ©n las respuestas necesarias en plot2*/
     /*kc, ti, y td son vectores, que contienen la ganancia
proporcional*/
     /*el tiempo integral y el tiempo derivativo*/
     /*se utilizan sus diferentes elementos para los diferentes
criterios*/

     switch (control) {

               case "p":
               double nump[1];
               double denp[1]={1};
               switch (metodo) {

                      case "zyn":
                       kc[0]=tau/(kp*tm);
                      nump[0]=kc[0];
```

```
                        control1.model("tf",nump,denp);
                        serie=control1.series(&sys);
                        retro.model("tf",numretro,denretro);
                        lazo=serie->feedback(&retro);
                        lazo->step(&plot2,NULL,NULL,NULL);

                        break;

                         case "lopez":


                         a[0]=1.410;
                        a[1]=0.9023;
                         a[2]=0.4897;
                         b[0]=-0.9170;
                         b[1]=-0.9850;
                        b[2]=-1.0850;

                        for(i=0;i<3;i++) {
                                kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
                                }

                        for(i=0;i<3;i++){
                                nump[0]=kc[i];
                                control1.model("tf",nump,denp);
                                serie=control1.series(&sys);
                                retro.model("tf",numretro,denretro);
                                lazo=serie->feedback(&retro);
                                lazo->step(&plot2,NULL,NULL,NULL);
                                }
                        plot2.legend("ISE",0);
                        plot2.legend("IAE",1);
                        plot2.legend("ITAE",2);

                        break;

                        default:
                         printf("El m&eacute;todo seleccionado no sintoniza
controladores tipo P\n");
                          break;

                        }
                 break;

                case "pi":

                double numpi[2];
                double denpi[2];
                switch (metodo) {
```

```
 case "zyn":
 kc[0]=0.9*tau/(kp*tm);
ti[0]=3.33*tm;
numpi[0]=kc[0]*ti[0];
numpi[1]=kc[0];
denpi[0]=ti[0];
denpi[1]=0;
control1.model("tf",numpi,denpi);
serie=control1.series(&sys);
retro.model("tf",numretro,denretro);
lazo=serie->feedback(&retro);
lazo->step(&plot2,NULL,NULL,NULL);

break;

case "lopez":

 a[0]=1.3050;
a[1]=0.9840;
 a[2]=0.8590;
 b[0]=-0.9600;
 b[1]=-0.9860;
b[2]=-0.9770;
c[0]=2.0325;
c[1]=1.6447;
 c[2]=1.4837;
 d[0]=-0.7390;
 d[1]=-0.7070;
d[2]=-0.6800;

for(i=0;i<3;i++) {
        kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
         ti[i]=c[i]*tau*pow(tm/tau, d[i]);
         }
for(i=0;i<3;i++){
        numpi[0]=kc[i]*ti[i];
        numpi[1]=kc[i];
        denpi[0]=ti[i];
        denpi[1]=0;
        control1.model("tf",numpi,denpi);
        serie=control1.series(&sys);
        retro.model("tf",numretro,denretro);
        lazo=serie->feedback(&retro);
        lazo->step(&plot2,NULL,NULL,NULL);
        }
plot2.legend("ISE",0);
plot2.legend("IAE",1);
plot2.legend("ITAE",2);


break;
```

```
case "rovira":

a[1]=0.7580;
 a[2]=0.5860;
b[1]=-0.8610;
b[2]=-0.9160;
c[1]=1.0200;
c[2]=1.0300;
d[1]=-0.3230;
d[2]=-0.1650;

for(i=1;i<3;i++) {
        kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
          ti[i]=tau/(c[i]+d[i]*(tm/tau));
          }
for(i=1;i<3;i++){
        numpi[0]=kc[i]*ti[i];
        numpi[1]=kc[i];
        denpi[0]=ti[i];
        denpi[1]=0;
        control1.model("tf",numpi,denpi);
        serie=control1.series(&sys);
        retro.model("tf",numretro,denretro);
        lazo=serie->feedback(&retro);
        lazo->step(&plot2,NULL,NULL,NULL);
        }
plot2.legend("IAE",0);
plot2.legend("ITAE",1);

break;


case "arrietareg":

a[1]=0.45;
a[2]=0.2607;
 b[1]=0.6494;
b[2]=0.6470;
 c[1]=-1.1251;
c[2]=-1.1055;
d[1]=-0.2551;
d[2]=-1.5926;
e[1]=1.8205;
e[2]=2.9191;
f[1]=0.4749;
f[2]=0.1789;


for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
          ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
          }
```

```
for(i=1;i<3;i++){
        numpi[0]=kc[i]*ti[i];
        numpi[1]=kc[i];
        denpi[0]=ti[i];
        denpi[1]=0;
        control1.model("tf",numpi,denpi);
        serie=control1.series(&sys);
        retro.model("tf",numretro,denretro);
        lazo=serie->feedback(&retro);
        lazo->step(&plot2,NULL,NULL,NULL);
        }
plot2.legend("IAE",0);
plot2.legend("ITAE",1);

break;


case "arrietaservo":
a[1]=0.2438;
a[2]=0.1140;
 b[1]=0.5305;
b[2]=0.5131;
 c[1]=-1.0299;
c[2]=-1.0382;
d[1]=0.9377;
d[2]=0.9953;
e[1]=0.4337;
e[2]=0.2073;
f[1]=0.8714;
f[2]=1.5246;


for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
          ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
          }

for(i=1;i<3;i++){
        numpi[0]=kc[i]*ti[i];
        numpi[1]=kc[i];
        denpi[0]=ti[i];
        denpi[1]=0;
        control1.model("tf",numpi,denpi);
        serie=control1.series(&sys);
        retro.model("tf",numretro,denretro);
        lazo=serie->feedback(&retro);
        lazo->step(&plot2,NULL,NULL,NULL);
        }
plot2.legend("IAE",0);
plot2.legend("ITAE",1);
break;
```

```
case "solerareg":
a[1]=0.1234;
a[2]=0;
 b[1]=1.0640;
b[2]=0.9239;
 c[1]=-0.9487;
c[2]=-0.8616;
d[1]=2.48;
d[2]=2.0230;
e[1]=0.0607;
e[2]=0.1336;
f[1]=2.0550;
f[2]=1.3590;


for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
          ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
          }

for(i=1;i<3;i++){
        numpi[0]=kc[i]*ti[i];
        numpi[1]=kc[i];
        denpi[0]=ti[i];
        denpi[1]=0;
        control1.model("tf",numpi,denpi);
        serie=control1.series(&sys);
        retro.model("tf",numretro,denretro);
        lazo=serie->feedback(&retro);
        lazo->step(&plot2,NULL,NULL,NULL);
        }
plot2.legend("IAE",0);
plot2.legend("ITAE",1);

break;

case "soleraservo":
if(tm/tau<=0.65){
    a[1]=0;
    b[1]=0.8629;
    c[1]=-0.5814;
    d[1]=0;
    e[1]=2.1360;
    f[1]=-0.0861;
}
else{
    a[1]=0;
    b[1]=0.8629;
    c[1]=-0.5817;
    d[1]=2.2440;
    e[1]=0.0555;
```

```
            f[1]=3.1930;
        }

        if(tm/tau<=0.35){
            a[2]=0;
            b[2]=0.6477;
            c[2]=-0.4927;
            d[2]=0;
            e[2]=1.7270;
            f[2]=-0.0429;
        }
        else{
            a[2]=0;
            b[2]=0.6477;
            c[2]=-0.4927;
            d[2]=1.7910;
            e[2]=0.1012;
            f[2]=1.772;
        }



        for(i=1;i<3;i++) {
                kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
                  ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
                  }

        for(i=1;i<3;i++){
                numpi[0]=kc[i]*ti[i];
                numpi[1]=kc[i];
                denpi[0]=ti[i];
                denpi[1]=0;
                control1.model("tf",numpi,denpi);
                serie=control1.series(&sys);
                retro.model("tf",numretro,denretro);
                lazo=serie->feedback(&retro);
                lazo->step(&plot2,NULL,NULL,NULL);
                }
        plot2.legend("IAE",0);
        plot2.legend("ITAE",1);

        break;


        }
break;

case "pid":
double numpid[3];
double denpid[3];
switch (metodo) {
```

```
case "zyn":
kc[0]=2*tau/(kp*tm);
 ti[0]=2.5*tm;
td[0]=0.4*tm;
numpid[0]=kc[0]*td[0]*ti[0]*(1+0.1);
numpid[1]=kc[0]*(ti[0]+0.1*td[0]);
numpid[2]=kc[0];
denpid[0]=0.1*td[0]*ti[0];
denpid[1]=ti[0];
denpid[2]=0;
control1.model("tf",numpid,denpid);
serie=control1.series(&sys);
retro.model("tf",numretro,denretro);
lazo=serie->feedback(&retro);
lazo->step(&plot2,NULL,NULL,NULL);

break;

case "lopez":
a[0]=1.4950;
a[1]=1.4350;
 a[2]=1.3570;
 b[0]=-0.9450;
 b[1]=-0.9210;
b[2]=-0.9470;
c[0]=0.9083;
c[1]=1.1390;
 c[2]=1.1876;
 d[0]=0.7710;
 d[1]=0.7490;
d[2]=0.7380;
e[0]=0.5600;
e[1]=0.4820;
e[2]=0.3810;
f[0]=1.0060;
f[1]=1.1370;
f[2]=0.9950;

for(i=0;i<3;i++) {
        kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
          ti[i]=c[i]*tau*pow(tm/tau, d[i]);
          td[i]=e[i]*tau*pow(tm/tau, f[i]);
          }
for(i=0;i<3;i++) {
        numpid[0]=kc[i]*td[i]*ti[i]*(1+0.1);
        numpid[1]=kc[i]*ti[i]+kc[i]*0.1*td[i];
        numpid[2]=kc[i];
        denpid[0]=0.1*td[i]*ti[i];
        denpid[1]=ti[i];
        denpid[2]=0;
        control1.model("tf",numpid,denpid);
```

```
        serie=control1.series(&sys);
        retro.model("tf",numretro,denretro);
        lazo=serie->feedback(&retro);
        lazo->step(&plot2,NULL,NULL,NULL);
        }
plot2.legend("ISE",0);
plot2.legend("IAE",1);
plot2.legend("ITAE",2);

break;

case "rovira":
a[1]=1.0860;
 a[2]=0.9650;
b[1]=-0.8690;
b[2]=-0.8500;
c[1]=0.7400;
c[2]=1.7960;
d[1]=-0.1300;
d[2]=-0.1465;
e[1]=0.3480;
e[2]=0.3080;
f[1]=0.9140;
f[2]=0.9290;
for(i=1;i<3;i++) {
        kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
          ti[i]=tau/(c[i]+d[i]*(tm/tau));
          td[i]=e[i]*tau*pow(tm/tau, b[i]);
          }
for(i=1;i<3;i++) {
        numpid[0]=kc[i]*td[i]*ti[i]*(1+0.1);
        numpid[1]=kc[i]*ti[i]+kc[i]*0.1*td[i];
        numpid[2]=kc[i];
        denpid[0]=0.1*td[i]*ti[i];
        denpid[1]=ti[i];
        denpid[2]=0;
        control1.model("tf",numpid,denpid);
        serie=control1.series(&sys);
        retro.model("tf",numretro,denretro);
        lazo=serie->feedback(&retro);
        lazo->step(&plot2,NULL,NULL,NULL);
        }

plot2.legend("IAE",0);
plot2.legend("ITAE",1);
break;

case "arrietareg":
 a[1]=0.45;
a[2]=0.2607;
 b[1]=0.6494;
b[2]=0.6470;
```

```
 c[1]=-1.1251;
c[2]=-1.1055;
d[1]=-0.2551;
d[2]=-1.5926;
e[1]=1.8205;
e[2]=2.9191;
f[1]=0.4749;
f[2]=0.1789;
g[1]=-0.0003;
g[2]=-0.0053;
h[1]=0.3838;
h[2]=0.3695;
j[1]=0.9479;
j[2]=0.9286;


for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
          ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
        td[i]=tau*(g[i]+h[i]*pow(tm/tau,j[i]));
          }

for(i=1;i<3;i++) {
        numpid[0]=kc[i]*td[i]*ti[i]*(1+0.1);
        numpid[1]=kc[i]*ti[i]+kc[i]*0.1*td[i];
        numpid[2]=kc[i];
        denpid[0]=0.1*td[i]*ti[i];
        denpid[1]=ti[i];
        denpid[2]=0;
        control1.model("tf",numpid,denpid);
        serie=control1.series(&sys);
        retro.model("tf",numretro,denretro);
        lazo=serie->feedback(&retro);
        lazo->step(&plot2,NULL,NULL,NULL);
        }

plot2.legend("IAE",0);
plot2.legend("ITAE",1);
break;

case "arrietaservo":

a[1]=0.2438;
a[2]=0.1140;
 b[1]=0.5305;
b[2]=0.5131;
 c[1]=-1.0299;
c[2]=-1.0382;
d[1]=0.9377;
d[2]=0.9953;
e[1]=0.4337;
e[2]=0.2073;
```

```
f[1]=0.8714;
f[2]=1.5246;
g[1]=-0.0146;
g[2]=-0.0169;
h[1]=0.3500;
h[2]=0.3126;
j[1]=0.8100;
j[2]=0.7417;


for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
          ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
        td[i]=tau*(g[i]+h[i]*pow(tm/tau,j[i]));
          }

for(i=1;i<3;i++) {
        numpid[0]=kc[i]*td[i]*ti[i]*(1+0.1);
        numpid[1]=kc[i]*ti[i]+kc[i]*0.1*td[i];
        numpid[2]=kc[i];
        denpid[0]=0.1*td[i]*ti[i];
        denpid[1]=ti[i];
        denpid[2]=0;
        control1.model("tf",numpid,denpid);
        serie=control1.series(&sys);
        retro.model("tf",numretro,denretro);
        lazo=serie->feedback(&retro);
        lazo->step(&plot2,NULL,NULL,NULL);
        }

plot2.legend("IAE",0);
plot2.legend("ITAE",1);
break;

case "solerareg":

a[1]=0.6199;
a[2]=0.5361;
 b[1]=1.1080;
b[2]=1.1210;
 c[1]=-1.4810;
c[2]=-1.4470;
d[1]=-1.1940;
d[2]=-2.9570;
e[1]=2.8340;
e[2]=4.7350;
f[1]=0.2686;
f[2]=0.1556;
g[1]=-0.2729;
g[2]=-0.2507;
h[1]=1.0640;
h[2]=0.9522;
```

```
j[1]=0.3521;
j[2]=0.3039;


for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
          ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
        td[i]=tau*(g[i]+h[i]*pow(tm/tau,j[i]));
          }

for(i=1;i<3;i++) {
        numpid[0]=kc[i]*td[i]*ti[i]*(1+0.1);
        numpid[1]=kc[i]*ti[i]+kc[i]*0.1*td[i];
        numpid[2]=kc[i];
        denpid[0]=0.1*td[i]*ti[i];
        denpid[1]=ti[i];
        denpid[2]=0;
        control1.model("tf",numpid,denpid);
        serie=control1.series(&sys);
        retro.model("tf",numretro,denretro);
        lazo=serie->feedback(&retro);
        lazo->step(&plot2,NULL,NULL,NULL);
        }

plot2.legend("IAE",0);
plot2.legend("ITAE",1);
break;

case "soleraservo":

if(tm/tau<=0.45){
    a[1]=0;
    b[1]=1.2460;
    c[1]=-0.8050;
    d[1]=2.1420;
    e[1]=-0.1356;
    f[1]=-0.2790;
}
else{
    a[1]=0;
    b[1]=1.2460;
    c[1]=-0.8050;
    d[1]=1.9630;
    e[1]=0.0805;
    f[1]=2.2110;
}

if(tm/tau<=0.65){
    a[2]=0.3004;
    b[2]=0.8471;
    c[2]=-0.9218;
    d[2]=1.6980;
```

```
            e[2]=0.3713;
            f[2]=0.2957;
        }
        else{
            a[2]=0.3004;
            b[2]=0.8471;
            c[2]=-0.9218;
            d[2]=1.9390;
            e[2]=0.1486;
            f[2]=1.593;
        }

        g[1]=0.4523;
        g[2]=0.4629;
        h[1]=0.2477;
        h[2]=0.1414;
        j[1]=0.9587;
        j[2]=1.275;


        for(i=1;i<3;i++) {
                kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
                  ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
                td[i]=tau*(g[i]+h[i]*pow(tm/tau,j[i]));
                  }
        for(i=1;i<3;i++) {
                numpid[0]=kc[i]*td[i]*ti[i]*(1+0.1);
                numpid[1]=kc[i]*ti[i]+kc[i]*0.1*td[i];
                numpid[2]=kc[i];
                denpid[0]=0.1*td[i]*ti[i];
                denpid[1]=ti[i];
                denpid[2]=0;
                control1.model("tf",numpid,denpid);
                serie=control1.series(&sys);
                retro.model("tf",numretro,denretro);
                lazo=serie->feedback(&retro);
                lazo->step(&plot2,NULL,NULL,NULL);
                }

        plot2.legend("IAE",0);
        plot2.legend("ITAE",1);


        break;




        }
break;
```

```
            }

    /*formato png para que sea interpretada por el browser*/
    plot2.outputType(PLOT_OUTPUTTYPE_STREAM, "png color");
    /*propiedades del gÅ•afico*/
    plot2.title("Servomecanismo");
    plot2.grid(PLOT_ON);
    plot2.size(0.7, 0.7);
    plot2.plotting();
    Response.end();


}
```

modulo5_image_2.ch

```
#!/bin/ch
#include <cgi.h>
#include <control.h>
#include  <chplot.h>
int main() {

  /*declaraciÃ³n de clases y variables*/
  int  i, k,np,nz,loc;
  chstrarray name[16],values[16], criterios[3];
  criterios[0]="ISE";
  criterios[1]="IAE";
  criterios[2]="ITAE";

  char *control, *modelo, *metodo;
  double num[5],den[5],y[2001],t[2001],pend[2000], kc[3]={0,0,0},
ti[3]={0,0,0},td[3]={0,0,0}, y25,t25,y50,t50,y75,t75,y63, t63,
m,p,tau,tm,kp,dt,tpm,ind, tfinal;
  double a[3]={0,0,0}, b[3]={0,0,0}, c[3]={0,0,0}, d[3]={0,0,0},
e[3]={0,0,0}, f[3]={0,0,0}, g[3]={0,0,0}, h[3]={0,0,0}, j[3]={0,0,0};
  double numretro[1]={1}, denretro[1]={1};
  class CResponse Response;
  class CRequest Request;
  class CServer Server;
  class CControl sys,control1,*serie, *lazo, sys4,retro;
  class CPlot plot,plot2;

  /*inicio y creaciÃ³n de respuesta tipo imagen*/
  Response.setContentType("image/png");
  Response.begin();

  /*limpia el buffer*/
  setbuf(stdout,NULL);


  /*obtenciÃ³n de parÃ¡metros modelo, control y metodo*/
  modelo=Request.getForm("modelo");
  control=Request.getForm("control");
  metodo=Request.getForm("metodo");

        switch(modelo){

        case "division":

        name[0]="num4";
         name[1]="num3";
         name[2]="num2";
         name[3]="num1";
         name[4]="num0";
         name[5]="den4";
```

```
name[6]="den3";
name[7]="den2";
name[8]="den1";
name[9]="den0";
name[10]="control";
name[11]="modelo";
name[12]="metodo";
values[0]=Request.getForm("num4");
values[1]=Request.getForm("num3");
values[2]=Request.getForm("num2");
values[3]=Request.getForm("num1");
values[4]=Request.getForm("num0");
values[5]=Request.getForm("den4");
values[6]=Request.getForm("den3");
values[7]=Request.getForm("den2");
values[8]=Request.getForm("den1");
values[9]=Request.getForm("den0");
values[10]=control;
values[11]=modelo;
values[12]=metodo;
break;

case "pyz":
name[0]="z1";
name[1]="z2";
name[2]="z3";
name[3]="z4";
name[4]="p1";
name[5]="p2";
name[6]="p3";
name[7]="p4";
name[8]="kp";
name[9]="modelo";
name[10]="control";
name[11]="nz";
name[12]="np";
name[13]="metodo";
values[0]=Request.getForm("z1");
values[1]=Request.getForm("z2");
values[2]=Request.getForm("z3");
values[3]=Request.getForm("z4");
values[4]=Request.getForm("p1");
values[5]=Request.getForm("p2");
values[6]=Request.getForm("p3");
values[7]=Request.getForm("p4");
values[8]=Request.getForm("kp");
values[9]=modelo;
values[10]=control;
values[11]=Request.getForm("nz");
values[12]=Request.getForm("np");
values[13]=metodo;
break;
```

```
    }


    /*creaciÃ³n de la funciÃ³n de transferencia de la planta*/
    /*segÃºn modelo seleccionado*/

    switch (modelo){

            case "division":

            for(k=0; k<5; k++) {
                    num[k]=atof(values[k]);
                    den[k]=atof(values[k+5]);
                    }
            sys.model("tf", num, den);
            break;

            case "pyz":

            nz=atoi(Request.getForm("nz"));
            array double complex zeros[nz];
            for(k=0; k<nz; k++) {
                    zeros[k]=atoc(values[k]);
                    }

            np=atoi(Request.getForm("np"));
            array double complex polos[np];
            for(k=0; k<np; k++) {
                    polos[k]=atoc(values[k+4]);
                    }

            kp=atof(values[8]);
            sys.model("zpk", zeros, polos, kp);
            break;
    }

/*curva de reacciÃ³n*/
sys.step(&plot,y,t,NULL);
tfinal=t[2000];
sys.step(&plot,y,t,NULL,tfinal*5);

/*cÃ¡lculo de tiempos para obtenciÃ³n de modelos*/

y63=y[2000]*0.63;
for(i=0;i < 2001; i++){
    loc=i;
    if(y[i] > y63){
            i=2001;
            }
    }
m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
p=y[loc]-m*t[loc];
```

```
t63=(y63-p)/m;

y25=y[2000]*0.25;
for(i=0;i < 2001; i++){
    loc=i;
    if(y[i] > y25){
            i=2001;
            }
    }
m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
p=y[loc]-m*t[loc];
t25=(y25-p)/m;



y75=y[2000]*0.75;
for(i=0;i < 2001; i++){
    loc=i;
    if(y[i] > y75){
            i=2001;
            }
    }
m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
p=y[loc]-m*t[loc];
t75=(y75-p)/m;

y50=y[2000]*0.50;
for(i=0;i < 2001; i++){
    loc=i;
    if(y[i] > y50){
            i=2001;
            }
    }
m=(y[loc]-y[loc-1])/(t[loc]-t[loc-1]);
p=y[loc]-m*t[loc];
t50=(y50-p)/m;

/*identificaciÃ³n de modelos segÃºn el caso*/
/*tangente para Zieglers y Nichols*/
/*polo doble usando 123c de Alfaro para Solera*/
/*primer orden usando 123c de Alfaro en los demÃ¡s casos*/

switch (metodo){
    case "zyn":
    kp=y[2000];
    tau=t63;
    /*rutina para identificar segun metodo de la tangente*/
    dt=t[1]-t[0];
    pend[0]=0;
    for(i=1;i<1999;i++){
            pend[i]=(y[i+1]-y[i-1])/(2*dt);
            }
```

```
        ind=maxloc(pend);

        tpm=t[ind];
        tm=(-1*y[ind])/pend[ind]+tpm;

        break;

        case "solerareg":
        kp=y[2000];
        tau=0.5776*(t75-t25);
        tm=1.5552*t25-0.5552*t75;
        if(tm<0)tm=0;

        break;

        case "soleraservo":
        kp=y[2000];
        tau=0.5776*(t75-t25);
        tm=1.5552*t25-0.5552*t75;
        if(tm<0)tm=0;
        break;

        default:
        kp=y[2000];
        tau=0.9102*(t75-t25);
        tm=1.2620*t25-0.2620*t75;
        if(tm<0)tm=0;
        break;
        }

      /*creaciÃ³n de la funciÃ³n de transferencia del controlador y
     conexiÃ³n como servomecanismo*/
     /*se crean tambiÃ©n las respuestas necesarias en plot2*/
     /*kc, ti, y td son vectores, que contienen la ganancia
proporcional*/
     /*el tiempo integral y el tiempo derivativo*/
     /*se utilizan sus diferentes elementos para los diferentes
criterios*/

      switch (control) {

               case "p":
               double nump[1];
               double denp[1]={1};
               switch (metodo) {

                       case "zyn":
                        kc[0]=tau/(kp*tm);
                       nump[0]=kc[0];
                       control1.model("tf",nump,denp);

                       lazo=sys.feedback(&control1);
```

```
                        lazo->step(&plot2,NULL,NULL,NULL);
                        break;

                         case "lopez":

                         a[0]=1.410;
                        a[1]=0.9023;
                         a[2]=0.4897;
                         b[0]=-0.9170;
                         b[1]=-0.9850;
                        b[2]=-1.0850;

                        for(i=0;i<3;i++) {
                                kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
                                        }
                        for(i=0;i<3;i++){
                                nump[0]=kc[i];
                                control1.model("tf",nump,denp);
                                lazo=sys.feedback(&control1);
                                lazo->step(&plot2,NULL,NULL,NULL);
                                }
                        plot2.legend("ISE",0);
                        plot2.legend("IAE",1);
                        plot2.legend("ITAE",2);

                        break;

                        default:
                         printf("El m&eacute;todo seleccionado no sintoniza
controladores tipo P\n");
                          break;

                        }
                 break;

                case "pi":

                double numpi[2];
                double denpi[2];
                switch (metodo) {



                         case "zyn":
                         kc[0]=0.9*tau/(kp*tm);
                        ti[0]=3.33*tm;
                        numpi[0]=kc[0]*ti[0];
                        numpi[1]=kc[0];
                        denpi[0]=ti[0];
                        denpi[1]=0;
                        control1.model("tf",numpi,denpi);
                        lazo=sys.feedback(&control1);
```

```
lazo->step(&plot2,NULL,NULL,NULL);

break;

case "lopez":
a[0]=1.3050;
a[1]=0.9840;
 a[2]=0.8590;
 b[0]=-0.9600;
 b[1]=-0.9860;
b[2]=-0.9770;
c[0]=2.0325;
c[1]=1.6447;
 c[2]=1.4837;
 d[0]=-0.7390;
 d[1]=-0.7070;
d[2]=-0.6800;
for(i=0;i<3;i++) {
        kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
          ti[i]=c[i]*tau*pow(tm/tau, d[i]);
          }
for(i=0;i<3;i++){
        numpi[0]=kc[i]*ti[i];
        numpi[1]=kc[i];
        denpi[0]=ti[i];
        denpi[1]=0;
        control1.model("tf",numpi,denpi);
        lazo=sys.feedback(&control1);
        lazo->step(&plot2,NULL,NULL,NULL);
        }
plot2.legend("ISE",0);
plot2.legend("IAE",1);
plot2.legend("ITAE",2);

break;

case "rovira":
a[1]=0.7580;
 a[2]=0.5860;
b[1]=-0.8610;
b[2]=-0.9160;
c[1]=1.0200;
c[2]=1.0300;
d[1]=-0.3230;
d[2]=-0.1650;

for(i=1;i<3;i++) {
        kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
          ti[i]=tau/(c[i]+d[i]*(tm/tau));
          }
for(i=1;i<3;i++){
        numpi[0]=kc[i]*ti[i];
```

```
        numpi[1]=kc[i];
        denpi[0]=ti[i];
        denpi[1]=0;
        control1.model("tf",numpi,denpi);
        lazo=sys.feedback(&control1);
        lazo->step(&plot2,NULL,NULL,NULL);
        }
plot2.legend("IAE",0);
plot2.legend("ITAE",1);


break;


case "arrietareg":
 a[1]=0.45;
a[2]=0.2607;
 b[1]=0.6494;
b[2]=0.6470;
 c[1]=-1.1251;
c[2]=-1.1055;
d[1]=-0.2551;
d[2]=-1.5926;
e[1]=1.8205;
e[2]=2.9191;
f[1]=0.4749;
f[2]=0.1789;


for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
         ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
         }
for(i=1;i<3;i++){
        numpi[0]=kc[i]*ti[i];
        numpi[1]=kc[i];
        denpi[0]=ti[i];
        denpi[1]=0;
        control1.model("tf",numpi,denpi);
        lazo=sys.feedback(&control1);
        lazo->step(&plot2,NULL,NULL,NULL);
        }
plot2.legend("IAE",0);
plot2.legend("ITAE",1);

break;


case "arrietaservo":
a[1]=0.2438;
a[2]=0.1140;
 b[1]=0.5305;
```

```
b[2]=0.5131;
 c[1]=-1.0299;
c[2]=-1.0382;
d[1]=0.9377;
d[2]=0.9953;
e[1]=0.4337;
e[2]=0.2073;
f[1]=0.8714;
f[2]=1.5246;


for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
          ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
          }
for(i=1;i<3;i++){
        numpi[0]=kc[i]*ti[i];
        numpi[1]=kc[i];
        denpi[0]=ti[i];
        denpi[1]=0;
        control1.model("tf",numpi,denpi);
        lazo=sys.feedback(&control1);
        lazo->step(&plot2,NULL,NULL,NULL);
        }
plot2.legend("IAE",0);
plot2.legend("ITAE",1);
break;

case "solerareg":
a[1]=0.1234;
a[2]=0;
 b[1]=1.0640;
b[2]=0.9239;
 c[1]=-0.9487;
c[2]=-0.8616;
d[1]=2.48;
d[2]=2.0230;
e[1]=0.0607;
e[2]=0.1336;
f[1]=2.0550;
f[2]=1.3590;


for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
          ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
          }
for(i=1;i<3;i++){
        numpi[0]=kc[i]*ti[i];
        numpi[1]=kc[i];
        denpi[0]=ti[i];
        denpi[1]=0;
```

```
        control1.model("tf",numpi,denpi);
        lazo=sys.feedback(&control1);
        lazo->step(&plot2,NULL,NULL,NULL);
        }
plot2.legend("IAE",0);
plot2.legend("ITAE",1);

break;

case "soleraservo":
if(tm/tau<=0.65){
    a[1]=0;
    b[1]=0.8629;
    c[1]=-0.5814;
    d[1]=0;
    e[1]=2.1360;
    f[1]=-0.0861;
}
else{
    a[1]=0;
    b[1]=0.8629;
    c[1]=-0.5817;
    d[1]=2.2440;
    e[1]=0.0555;
    f[1]=3.1930;
}

if(tm/tau<=0.35){
    a[2]=0;
    b[2]=0.6477;
    c[2]=-0.4927;
    d[2]=0;
    e[2]=1.7270;
    f[2]=-0.0429;
}
else{
    a[2]=0;
    b[2]=0.6477;
    c[2]=-0.4927;
    d[2]=1.7910;
    e[2]=0.1012;
    f[2]=1.772;
}



for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
         ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
         }
for(i=1;i<3;i++){
```

```
                numpi[0]=kc[i]*ti[i];
                numpi[1]=kc[i];
                denpi[0]=ti[i];
                denpi[1]=0;
                control1.model("tf",numpi,denpi);
                lazo=sys.feedback(&control1);
                lazo->step(&plot2,NULL,NULL,NULL);
                }
        plot2.legend("IAE",0);
        plot2.legend("ITAE",1);
        break;


        }
break;

case "pid":
double numpid[3];
double denpid[3];
switch (metodo) {

        case "zyn":
        kc[0]=2*tau/(kp*tm);
         ti[0]=2.5*tm;
        td[0]=0.4*tm;
        numpid[0]=kc[0]*td[0]*ti[0]*(1+0.1);
        numpid[1]=kc[0]*(ti[0]+0.1*td[0]);
        numpid[2]=kc[0];
        denpid[0]=0.1*td[0]*ti[0];
        denpid[1]=ti[0];
        denpid[2]=0;
        control1.model("tf",numpid,denpid);
        lazo=sys.feedback(&control1);
        lazo->step(&plot2,NULL,NULL,NULL);

        break;

        case "lopez":
        a[0]=1.4950;
        a[1]=1.4350;
         a[2]=1.3570;
         b[0]=-0.9450;
         b[1]=-0.9210;
        b[2]=-0.9470;
        c[0]=0.9083;
        c[1]=1.1390;
         c[2]=1.1876;
         d[0]=0.7710;
         d[1]=0.7490;
        d[2]=0.7380;
        e[0]=0.5600;
        e[1]=0.4820;
```

```
e[2]=0.3810;
f[0]=1.0060;
f[1]=1.1370;
f[2]=0.9950;

for(i=0;i<3;i++) {
        kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
          ti[i]=c[i]*tau*pow(tm/tau, d[i]);
          td[i]=e[i]*tau*pow(tm/tau, f[i]);
          }
for(i=0;i<3;i++) {
        numpid[0]=kc[i]*td[i]*ti[i]*(1+0.1);
        numpid[1]=kc[i]*ti[i]+kc[i]*0.1*td[i];
        numpid[2]=kc[i];
        denpid[0]=0.1*td[i]*ti[i];
        denpid[1]=ti[i];
        denpid[2]=0;
        control1.model("tf",numpid,denpid);
        lazo=sys.feedback(&control1);
        lazo->step(&plot2,NULL,NULL,NULL);
        }
plot2.legend("ISE",0);
plot2.legend("IAE",1);
plot2.legend("ITAE",2);


break;

case "rovira":
a[1]=1.0860;
 a[2]=0.9650;
b[1]=-0.8690;
b[2]=-0.8500;
c[1]=0.7400;
c[2]=1.7960;
d[1]=-0.1300;
d[2]=-0.1465;
e[1]=0.3480;
e[2]=0.3080;
f[1]=0.9140;
f[2]=0.9290;

for(i=1;i<3;i++) {
        kc[i]=(a[i]/kp)*pow(tm/tau, b[i]);
          ti[i]=tau/(c[i]+d[i]*(tm/tau));
          td[i]=e[i]*tau*pow(tm/tau, b[i]);
          }
for(i=1;i<3;i++) {
        numpid[0]=kc[i]*td[i]*ti[i]*(1+0.1);
        numpid[1]=kc[i]*ti[i]+kc[i]*0.1*td[i];
        numpid[2]=kc[i];
        denpid[0]=0.1*td[i]*ti[i];
```

```
        denpid[1]=ti[i];
        denpid[2]=0;
        control1.model("tf",numpid,denpid);
        lazo=sys.feedback(&control1);
        lazo->step(&plot2,NULL,NULL,NULL);
        }

plot2.legend("IAE",0);
plot2.legend("ITAE",1);

break;

case "arrietareg":
 a[1]=0.45;
a[2]=0.2607;
 b[1]=0.6494;
b[2]=0.6470;
 c[1]=-1.1251;
c[2]=-1.1055;
d[1]=-0.2551;
d[2]=-1.5926;
e[1]=1.8205;
e[2]=2.9191;
f[1]=0.4749;
f[2]=0.1789;
g[1]=-0.0003;
g[2]=-0.0053;
h[1]=0.3838;
h[2]=0.3695;
j[1]=0.9479;
j[2]=0.9286;


for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
          ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
        td[i]=tau*(g[i]+h[i]*pow(tm/tau,j[i]));
          }
for(i=1;i<3;i++) {
        numpid[0]=kc[i]*td[i]*ti[i]*(1+0.1);
        numpid[1]=kc[i]*ti[i]+kc[i]*0.1*td[i];
        numpid[2]=kc[i];
        denpid[0]=0.1*td[i]*ti[i];
        denpid[1]=ti[i];
        denpid[2]=0;
        control1.model("tf",numpid,denpid);
        lazo=sys.feedback(&control1);
        lazo->step(&plot2,NULL,NULL,NULL);
        }

plot2.legend("IAE",0);
plot2.legend("ITAE",1);
```

```
break;

case "arrietaservo":
a[1]=0.2438;
a[2]=0.1140;
 b[1]=0.5305;
b[2]=0.5131;
 c[1]=-1.0299;
c[2]=-1.0382;
d[1]=0.9377;
d[2]=0.9953;
e[1]=0.4337;
e[2]=0.2073;
f[1]=0.8714;
f[2]=1.5246;
g[1]=-0.0146;
g[2]=-0.0169;
h[1]=0.3500;
h[2]=0.3126;
j[1]=0.8100;
j[2]=0.7417;


for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
          ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
        td[i]=tau*(g[i]+h[i]*pow(tm/tau,j[i]));
          }
for(i=1;i<3;i++) {
        numpid[0]=kc[i]*td[i]*ti[i]*(1+0.1);
        numpid[1]=kc[i]*ti[i]+kc[i]*0.1*td[i];
        numpid[2]=kc[i];
        denpid[0]=0.1*td[i]*ti[i];
        denpid[1]=ti[i];
        denpid[2]=0;
        control1.model("tf",numpid,denpid);
        lazo=sys.feedback(&control1);
        lazo->step(&plot2,NULL,NULL,NULL);
        }

plot2.legend("IAE",0);
plot2.legend("ITAE",1);


break;

case "solerareg":
a[1]=0.6199;
a[2]=0.5361;
 b[1]=1.1080;
```

```
b[2]=1.1210;
 c[1]=-1.4810;
c[2]=-1.4470;
d[1]=-1.1940;
d[2]=-2.9570;
e[1]=2.8340;
e[2]=4.7350;
f[1]=0.2686;
f[2]=0.1556;
g[1]=-0.2729;
g[2]=-0.2507;
h[1]=1.0640;
h[2]=0.9522;
j[1]=0.3521;
j[2]=0.3039;


for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
          ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
        td[i]=tau*(g[i]+h[i]*pow(tm/tau,j[i]));
          }

for(i=1;i<3;i++) {
        numpid[0]=kc[i]*td[i]*ti[i]*(1+0.1);
        numpid[1]=kc[i]*ti[i]+kc[i]*0.1*td[i];
        numpid[2]=kc[i];
        denpid[0]=0.1*td[i]*ti[i];
        denpid[1]=ti[i];
        denpid[2]=0;
        control1.model("tf",numpid,denpid);
        lazo=sys.feedback(&control1);
        lazo->step(&plot2,NULL,NULL,NULL);
        }

plot2.legend("IAE",0);
plot2.legend("ITAE",1);


break;

case "soleraservo":

if(tm/tau<=0.45){
    a[1]=0;
    b[1]=1.2460;
    c[1]=-0.8050;
    d[1]=2.1420;
    e[1]=-0.1356;
    f[1]=-0.2790;
}
else{
```

```
    a[1]=0;
    b[1]=1.2460;
    c[1]=-0.8050;
    d[1]=1.9630;
    e[1]=0.0805;
    f[1]=2.2110;
}

if(tm/tau<=0.65){
    a[2]=0.3004;
    b[2]=0.8471;
    c[2]=-0.9218;
    d[2]=1.6980;
    e[2]=0.3713;
    f[2]=0.2957;
}
else{
    a[2]=0.3004;
    b[2]=0.8471;
    c[2]=-0.9218;
    d[2]=1.9390;
    e[2]=0.1486;
    f[2]=1.593;
}

g[1]=0.4523;
g[2]=0.4629;
h[1]=0.2477;
h[2]=0.1414;
j[1]=0.9587;
j[2]=1.275;


for(i=1;i<3;i++) {
        kc[i]=(a[i]+b[i]*pow(tm/tau, c[i]))/kp;
          ti[i]=tau*(d[i]+e[i]*pow(tm/tau,f[i]));
        td[i]=tau*(g[i]+h[i]*pow(tm/tau,j[i]));
          }
for(i=1;i<3;i++) {
        numpid[0]=kc[i]*td[i]*ti[i]*(1+0.1);
        numpid[1]=kc[i]*ti[i]+kc[i]*0.1*td[i];
        numpid[2]=kc[i];
        denpid[0]=0.1*td[i]*ti[i];
        denpid[1]=ti[i];
        denpid[2]=0;
        control1.model("tf",numpid,denpid);
        lazo=sys.feedback(&control1);
        lazo->step(&plot2,NULL,NULL,NULL);
        }

plot2.legend("IAE",0);
plot2.legend("ITAE",1);
```

```
                        break;



                        }
                break;


                }
  /*formato png para que sea interpretada por el browser*/
  plot2.outputType(PLOT_OUTPUTTYPE_STREAM, "png color");
  /*propiedades del gÅ•afico*/
  plot2.title("Regulador");
  plot2.grid(PLOT_ON);
  plot2.size(0.7,0.7);
  plot2.plotting();
  Response.end();
}
```